Sensor Fusion & Fault Diagnosis via Bayesian Model Selection

A. What is this project about?

Theme. Three sensors measure the same (time-varying) physical quantity. Occasionally, one sensor becomes faulty (bias or spikes). You will (i) detect which sensor is faulty in real time and (ii) still estimate the true signal by fusing sensor data.

Core skills gained

- Modeling random signals (state-space, random walk/AR(1), Gaussian noise)
- Likelihoods, priors, posteriors; Bayesian model selection
- Online inference using the Kalman filter (innovation likelihood / log-evidence)
- Model averaging for robust sensor fusion
- Practical fault flags, thresholds, ROC thinking
- (Stretch) HMM for fault persistence + Viterbi vs Bayesian filtering

Tools. MATLAB or Python (NumPy/SciPy). No external data needed—**You simulate your own data**.

B. Project details

B1. Generative model (baseline)

- **Hidden state (true signal)**
  Random walk: $x_t = x_{t-1} + w_t, \quad w_t \sim \mathcal{N}(0, q)$.
  (Option: AR(1): $x_t = ax_{t-1} + w_t$ with $|a| < 1$.)
- **Sensors (i = 1,2,3)**
  Healthy: $y_{i,t} = x_t + v_{i,t}, \quad v_{i,t} \sim \mathcal{N}(0, r_i)$ with distinct r_i.
  Faulty (bias): $y_{i,t} = x_t + b + v_{i,t}$, where $b \sim \mathcal{N}(\mu_b, \sigma_b^2)$ (fixed per fault episode).
  Faulty (spike): with probability p_{spike}, add $s_t \sim \mathcal{N}(0, \sigma_s^2)$ to the healthy model.
- **Fault prior (simple, i.i.d. per t):** with prob p_f one sensor is faulty, chosen uniformly; else none faulty.
  Models: $\mathcal{M}_0$ (no fault), $\mathcal{M}_1$ ("sensor 1 faulty"), $\mathcal{M}_2, \mathcal{M}_3$.

Parameters for you to explore:

$$\begin{aligned} q &\in [10^{-4}, 10^{-2}], \\ r_1 &< r_2 < r_3 \text{ (e.g., } 0.01, 0.04, 0.09), \\ p_f &\in [0.05, 0.2], \\ \text{bias } b &\sim \mathcal{N}(0,1) \text{ or fixed } b = 1, \\ \text{spike } \sigma_s^2 &\in [0.5, 4]. \end{aligned}$$

B2. Likelihood and Bayesian model selection

Let $y_t = [y_{1,t}, y_{2,t}, y_{3,t}]^\top$. For each model $\mathcal{M}_j$ (which specifies which sensor, if any, is faulty and how), define an observation matrix $H^{(j)}$ and noise covariance $R_t^{(j)}$. For the **bias-fault case**, augment the state to include a (piecewise constant) bias term for the faulty sensor:

$$\underbrace{\begin{bmatrix} x_t \\ b_t \end{bmatrix}}_{z_t} = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{F} \underbrace{\begin{bmatrix} x_{t-1} \\ b_{t-1} \end{bmatrix}}_{z_{t-1}} + \underbrace{\begin{bmatrix} w_t \\ \eta_t \end{bmatrix}}_{\tilde{w}_t}, \quad \eta_t \sim \mathcal{N}(0, \epsilon) \text{ (small to keep } b_t \text{ nearly constant)}$$

and

$$y_t = H^{(j)} z_t + v_t, \quad v_t \sim \mathcal{N}(0, R_t^{(j)}).$$

- For $\mathcal{M}_0$ (no fault): $z_t = x_t$, $H^{(0)} = [1, 1, 1]^\top$ stacked appropriately, $R^{(0)} = \mathrm{diag}(r_1, r_2, r_3)$.
- For $\mathcal{M}_1$ (sensor 1 biased): $z_t = [x_t, b_t]$, observation for sensor 1 is $x_t + b_t$; sensors 2–3 observe x_t only.
  Spikes can be modeled by inflating the faulty sensor's variance at times with spikes: $R_{t,ii}^{(j)} = r_i + \sigma_s^2$.

Online marginal likelihood ("evidence").
Run a (possibly augmented-state) Kalman filter per model $\mathcal{M}_j$. At each step, compute the innovation and its covariance:

$$\tilde{y}_t^{(j)} = y_t - H^{(j)} \hat{z}_{t|t-1}^{(j)}, \quad S_t^{(j)} = H^{(j)} P_{t|t-1}^{(j)} H^{(j)\top} + R_t^{(j)}.$$

The one-step predictive likelihood is

$$p(y_t \mid y_{1:t-1}, \mathcal{M}_j) = \mathcal{N}\left(\tilde{y}_t^{(j)}; 0, S_t^{(j)}\right).$$

Accumulate log-evidence:

$$\log p(y_{1:t} \mid \mathcal{M}_j) = \sum_{\tau=1}^{t} \log \mathcal{N}\left(\tilde{y}_\tau^{(j)}; 0, S_\tau^{(j)}\right).$$

Posterior over models (recursive):

$$P(\mathcal{M}_j \mid y_{1:t}) \propto P(\mathcal{M}_j)\, p(y_t \mid y_{1:t-1}, \mathcal{M}_j)\, P(\mathcal{M}_j \mid y_{1:t-1}),$$

then normalize over $j \in \{0, 1, 2, 3\}$.

B3. Fusion by Bayesian model averaging

Each model's filter yields a state estimate $\hat{x}_{t|t}^{(j)}$ (extract x_t from $\hat{z}_t$). The fused estimate:

$$\hat{x}_t^{(\mathrm{BMA})} = \sum_j P(\mathcal{M}_j \mid y_{1:t})\, \hat{x}_{t|t}^{(j)}.$$

Uncertainty can be combined via mixture-of-Gaussians second-moment:

$$P_t^{(\mathrm{BMA})} = \sum_j P(\mathcal{M}_j \mid y_{1:t}) \Big(P_{t|t}^{(j)} + (\hat{x}_{t|t}^{(j)} - \hat{x}_t^{(\mathrm{BMA})})(\cdot)^\top \Big).$$

B4. Fault flagging

- **Hard decision:** flag sensor i as faulty if $P(\mathcal{M}_i \mid y_{1:t}) > \tau$ (e.g., $\tau = 0.8$).
- **Soft view:** plot the full posterior over $\{\mathcal{M}_j\}$ over time and compare to ground truth fault times.

B5. Stretch goal: HMM prior over faults

Model a 4-state Markov chain for the model index $M_t \in \{\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3\}$ with transition matrix A (high self-persistence). Then:

$$P(M_t = j \mid y_{1:t}) \propto \underbrace{\sum_k A_{kj} P(M_{t-1} = k \mid y_{1:t-1})}_{\text{prediction}} \times p(y_t \mid y_{1:t-1}, \mathcal{M}_j).$$

- **Filtering:** Forward recursion for $P(M_t \mid y_{1:t})$.
- **Most likely fault sequence:** Viterbi on $\log A$ and $\log p(y_t \mid \mathcal{M}_j)$.
- **State estimate:** Either BMA with $P(M_t \mid y_{1:t})$ or run the KF conditioned on the Viterbi path.

C. Simulation & experiments students should run

1. **Generate data**
   - $T = 500$–1000; choose q, r_i.
   - Introduce one fault episode (e.g., $t \in [200, 320]$) on a chosen sensor: bias b or spikes.
   - (Option) Multiple episodes; change which sensor is faulty.
2. **Implement per-model filters**
   - $\mathcal{M}_0$: standard KF on x_t.
   - $\mathcal{M}_i$: augmented-state KF $[x_t, b_t]$ or variance inflation for spikes.
3. **Online posteriors & fusion**

- Update $P(\mathcal{M}_j \mid y_{1:t})$ each step.
- Plot $\hat{x}_t^{(\mathrm{BMA})}$ vs. true x_t; report RMSE.
- Plot posteriors over models vs. time; show fault flag and ground truth.

4. **Ablations / sensitivity**
   - Vary p_f, q, r_i, σ_s^2.
   - Compare **best single sensor**, **naïve average**, **oracle (knows fault)**, **BMA**.
5. **Stretch (HMM)**
   - Choose A with strong diagonal (e.g., 0.98 stay, 0.02 spread).
   - Compare Viterbi vs. Bayesian filtering fault timelines.

D. What to hand in (deliverables)

- **Brief report (4–6 pages):** problem setup, models, equations, algorithm, experiments, plots, discussion.
- **Code:** clean, commented; seed for reproducibility.
- **Plots (minimum):**
  - (i) true x_t vs. $\hat{x}_t^{(\mathrm{BMA})}$;
  - (ii) model posteriors over time;
  - (iii) per-sensor residuals/innovations;
  - (iv) bar chart of RMSE across methods.
- **Readme:** how to run; parameter choices.

F. Pseudo-code

```
Initialize priors: P(M_j)=1/4, KF_j state mean/cov for each model j
for t = 1..T:
  observe y_t (3x1)

  for each model j in {0,1,2,3}:
    # Predict
    xhat_pred_j, P_pred_j = KF_predict_j()
    # Innovation and evidence
    innov_j = y_t - H_j * xhat_pred_j
    S_j = H_j * P_pred_j * H_j' + R_j_t
    ell_j = N(innov_j; 0, S_j)
    # Update
    K_j = P_pred_j * H_j' * inv(S_j)
```

```
    xhat_filt_j = xhat_pred_j + K_j * innov_j
    P_filt_j = (I - K_j * H_j) * P_pred_j

    store xhat_filt_j, P_filt_j, ell_j

  # Model posterior update
  post_j ~ ell_j * prior_j
  normalize post_j
  prior_j <- post_j

  # Bayesian model averaging
  xhat_BMA_t = sum_j post_j * extract_x(xhat_filt_j)

  # Fault flag
  if max_j post_j > tau: flag argmax_j post_j
end
```

G. Starter parameter set (easy mode)

$$q = 10^{-3},\ r_1 = 0.01,\ r_2 = 0.04,\ r_3 = 0.09$$

Bias fault on sensor 2, $b = +1.0$ for $t \in [200, 320]$
Spike mode alternative: $p_{\text{spike}} = 0.2,\ \sigma_s^2 = 1.0$ during the episode
Prior $P(\mathcal{M}_j) = 1/4$, threshold $\tau = 0.8$

H. FAQ

- **Do you need external data?** No—simulate everything.
- **How do you choose KF dimensions for bias faults?** Augment the state with a nearly constant bias b_t (very small process noise ϵ).
- **How do you compute model evidence?** Use the KF **innovation likelihood** at each step and multiply (sum logs) over time.
- **What if two sensors go bad?** Out of scope—stick to "at most one faulty" (or document as limitation).
- **How to set thresholds?** Empirically via validation runs; plot posteriors and pick τ that trades false alarms vs misses.