# TTK4250 Sensor Fusion

# Assignment 5

**Result producing code hand in (5%):** *Wednesday 14. October 16.00* on Blackboard as a single `.zip` file.
**Report hand in (10%):** *Friday 16. October 16.00* on Blackboard as a single PDF file.

---

**Pairs:** This assignment is to be solved in pairs. These pairs will be the same for all the graded assignments.

**Code hand in:** Functioning python code that produces your results for task 2. This will account for one third of the possible points for this assignment (5% of total grade).

**Report hand in:** A 4 page (in total!) report is to be handed in by 16.October 16.00 as a PDF on Blackboard. Points will be withdrawn for reports longer than 4 pages. As the report is quite short, you should *consider carefully* what to have in it.

**Report content:** The algorithms are given in the book, so you can simply refer to the book when what you want to say is already stated there. For task 2 and 3, we want to see

- plot of the total, position and velocity NEES along with your chosen confidence bounds for these,
- plot of estimation error (Euclidian distance) in position and velocity for the tracker run with your chosen parameters
- plot of the mode probabilities,
- the numbers of the averaged NEESes along with their confidence bounds,
- the number of times the NEESes fall within the confidence regions
- and the positional and velocity RMSE (mean taken over time)

for a selected parameter set. We then expect you to discuss what these number and parameters means for the tracker in practice, and what is gained or lost by changing the parameters and why. Points can also be given for other plots, numbers and considerations, but what to show is *up to you* to decide.

---

## Note:

This is not supposed to be a "try one million parameter sets" assignment. You are supposed to show your understanding of the algorithms in theory and practice. Yes, you do have to test a fair share of parameters, but doing just that will not give you the best grade. This is supposed to be an engineering/scientific assignment, where in some sense you as a consultant have the task of showing a company what they can get from these algorithms on their datasets. That is, making sure that the algorithm is correct (you can get help with that), making hypotheses to test, selecting which parameters to try in a reasonable fashion, and documenting your findings in terms of consistency, the stated metrics and the estimated trajectory, is what will give you a good result. Any theoretical or practical insight from the lectures or book elaborated upon or applied to the data sets also counts positive.

**Task 1:** *Make sure the PDA class is general*

This task is worth up to 3% for the code hand in. Full score is given if the classes are capable of doing the job of the EKF, IMM, PDA and IMM-PDA.

You are to make sure that the PDA from assignment 5 can operate with both EKF and IMM, so this will reuse and extend the PDA class you made in assignment 5. This means that the different methods of the PDA class must be able to handle both input data for IMM and EKF.Make sure that all functionality that is specialized to a certain state estimator is implemented by the state estimator and not the PDA class. The class skeletons to use for this assignment can be found on Blackboard under assignments 3–5.

(a) Gating: We shall adopt the rule that the track gates a measurement if at least one of the modes gates this measurement, you are free to investigate other options if wanted.

*Hint*: IMM.NIS should provide some of what you need. Also, the function `np.any` might come in handy.

(b) You need to implement the log-likelihood function for the IMM class. It can be found as the logarithm of equation (7.55) in the book.

(c) `IMM.reduce_mixture` is probably the most complicated to implement. We have the probabilities $\Pr(s_k = i|a_k = j, Z_{1:k})$ and $\Pr(a_k = j|Z_{1:k})$, as well as the densities $f(x_k|a_k = j, s_k = i, Z_{1:k})$. What we want is the probabilities

$$\Pr(s_k = i|Z_{1:k}) = \sum_{j=0}^{m_k} \Pr(s_k = i|a_k = j, Z_{1:k}) \Pr(a_k = j|Z_{1:k}) \tag{1}$$

(the total probability theorem), as well as the densities

$$f(x_k|s_k = i, Z_{1:k}) = \sum_{j=0}^{m_k} f(x_k|a_k = j, s_k = i, Z_{1:k}) \Pr(a_k = j|s_k = i, Z_{1:k}) \tag{2}$$

$$= \sum_{j=0}^{m_k} f(x_k|a_k = j, s_k = i, Z_{1:k}) \frac{\Pr(s_k = i|a_k = j, Z_{1:k}) \Pr(a_k = j|Z_{1:k})}{\Pr(s_k = i|Z_{1:k})} \tag{3}$$

(first the total probability theorem and then Bayes on the probabilities). As usual we approximate $f(x_k|s_k = i, Z_{1:k})$ by a single Gaussian that matches the first two moments.

For this step you are given a mixture of a mixture. Ie. `immstate_mixture.weights` are $\Pr(a_k|Z_{1:k})$ and `immstate_mixture.components` are $f(\mathbf{y}_k|a_k, Z_{1:k})$ found in equations (7.47) and (7.49) in the book. This means that `immstate_mixture.components[j]` contains the result of equations (7.51) and (7.52) for $a_k = j$ as its components and weights respectively.

**Task 2:** *Tune the IMM to the given data*

This task is worth up to 2% for the code hand in, and up to 4% in the report. Full score is given in the code/result hand in for a reasonable (not necessarily optimal) choice of parameters that works on the data set. In the report you should do a bit more analysis. To achieve a full score we also want to see at least some minor reflections on the algorithm and the approximations it does in light of your results.

You are given a data set on Blackboard in the file `data_for_imm_pda.mat`. This corresponds to the same scenario as in assignment 4, but with added false alarms and missed detections removed. It contains

- `K: int` (The (positive) number of time steps)
- `Ts: float` (The (positive) sampling time)
- `Xgt: np.ndarray, shape=(K, 5)` (the state ground truth, where the state order is x-y position, x-y velocities and turn rate)
- `Z: List[np.ndarray], len(Z) == K, Z[k].shape == (m_k, 2)` (a list of measurements so that the $k^{\text{th}}$ element are the $m_k$ measurements at time step $k$).

- `true_association: np.ndarray, shape=(K,)` (the true association for each time step, ie. $a_k$)

Tune the IMM-PDAF, using a CV model and a CT model with position measurements to this dataset. You need to tune the process disturbances of the models, the measurement noise, the detection probability, the false alarm intensity and the gate size.

*Hint*: Since this is the same scenario as in assignment 3, you should already have a good starting point for the process noises and measurement noises, as well as means to investigate the tuning of these parameters without the problems that false alarms and missed detections introduce.

**Task 3:** *IMM-PDAF on the real radar data set "Joyride"*

This task is worth up to 6% in the report hand in. To achieve a full score we also want to see at least some minor reflections on the algorithm assumptions and the approximations it does, in light of your results in this data set. Relating results in this task to the previous assignments and tasks *can* give additional points.

You are given a data set on Blackboard in the file `data_joyridedata.mat`. This corresponds to the real world "Joyride" dataset partially described in the book and lecture slides. It is collected with a RADAR mounted on a moving boat in Trondheimsfjorden, where a quite agile boat that we want to track is doing a "joyful ride" inside the field of view of the RADAR.The RADAR measurements have been processed into single points in cartesian coordinates per RADAR scan. The dataset consists of

- K (=200): The number of time steps in the data set
- time (200 floats): the time when the measurements were captured
- Ts (199 floats): the sampling time between the measurements (calculated using `np.diff(time)`)
- Xgt (200 x 4 np.ndarray): "ground truth" as measured by GPS.The states are x-y position followed by x-y speed. Note that this is not necessarily a perfect ground truth.
- ownship (200 x 4 np.ndarray): the RADAR position and speed at each time step. Same layout as for Xgt.
- Z (list of length 200 containing m_k x 2 np.ndarray): a list of measurement so that Z[k] is the m_k measurements at time step k.

Tune your PDAF and/or IMM-PDAF to this data set. Experiment with using a EKF (CV and CT) and IMM (CV-CT and CV-CT-CVhigh) as your state estimators. This is a challenging data set, and we have not been able to get some of these model configurations to work satisfactory (see section 7.6.2).

Hint: A reasonable, not necessarily optimal, value for the measurement noise standard deviation is found to be $\sigma_a = 6$, but also $\sigma_a = 10$ can give good results.