Introduction to Estimation and the Kalman Filter (KC-1)
Extended Kalman Filter Navigation System Example Documentation

The following Matlab code implements the extended Kalman filter navigation system example described in Section 8 of the course notes. The code is divided into two main parts:

1. A simulation that generates a vehicle trajectory and observations of landmarks,
2. An extended Kalman filter which takes as input the control and observation inputs generated during the simulation and estimates vehicle position, orientation and effective wheel radius.

To run the code:

1. Start up Matlab and change to the directory containing the source code.
2. Run the script file “set_up”. This sets up a graphical window that allows input of beacons and vehicle trajectory. Follow the instructions as printed in the Matlab command window. Note, the trajectory generation function uses the Matlab “spline” commands, so trajectories that loop back (have more than one y value for a given x value) are not allowed. The function “set_up” uses the trajectory input to compute control inputs for the vehicle to maintain track. These are used later in the estimation cycle.
3. Run the script file “run_obs”. This simulates the operation of a scanning radar mounted on the vehicle and generates simulated observation information to drive the estimator.
4. Run the script file “run_filt”. This runs the filter itself, using data from control inputs and observations. The run_filt script file also initialises state and state covariance.
5. Run the script file “plots”. This plots out the results of the filtering process in the form found in Section 8 of the notes.
6. The file “ginit” contains all global definitions for variances, vehicle parameters, etc.

The code is initially set up to run a conventional and well tuned filter. The objective of the laboratory is:

- To see how the EKF is coded and how it works.
- To understand the process of tuning and operating an EKF.
- To see the effects of faults and changes in initial conditions.

The filtering code structure is reasonably generic. It simply calls two functions to predict (using a platform model) and then update (using an observation model), the state and state covariance. It should not be hard to modify these for other filtering problems.

Function Definitions:

a=a_add(a1,a2):
Adds two angles so that the result always remains between +/- pi

a=a_sub(a1,a2):
Subtracts two angles so that the result always remains between +/- pi

beacons-get_beacons:
A function to graphically input beacon locations beacons.

[xnext,unext]=get_control(x,u,perr,oerr):
A function to compute a new control vector for the vehicle and to do a one step vehicle simulation with this vector

[perr,oerr,index]=get_err(x,path):
A function to compute the position and orientation error of the vehicle with respect to a path.

path=get_path(beacons):
A function to graphically acquire spline points and compute a spline fit path for vehicle.

ginit:
Values for global variables

globals:
Definitions of global variables. Must be called in every function.

[xest,Pest,xpred,Ppred,innov,innvar]=
kfilter(obs,u,xinit,Pinit,beacons):
The main Kalman filter for the navigation system. Takes in a sequence of observations and control inputs, together with initial conditions, and produces state estimates, state predictions and innovations.

obs=obs_seq(state,beacons):
A function to gather a complete observation sequence from a true path and list of beacons.

onestep:
A script file allowing a single step in the filter to be executed. This allows each part of the calculation to be understood in detail. Requires complete filter to have been run previously

[obs_p,state_p]=p_obs(obs,state):
Afunction to place vehicle centered observations in global coordinates for checking of consistency

plots
A script file for convenient plotting of filter results.

[xpred, Ppred]=pred(xest,Pest,dt,u):
Function to generate a one-step vehicle prediction from previous estimate and control input.
[binnov, binnovsig]=proc_innov(innov,innovar,obs,bnum):
A function to process innovation results, extracting (optionally) innovations for specific beacons and standard deviations.

[obs,b]=rad_sim(start_loc,end_loc,beacons):
A function to simulate radar observations takes a start and end location for the vehicle, the vehicle parameter set and a map of beacons. Returns an observation if one is made during this time slot.

replot:
Replots vehicle trajectory and beacon location information.

run_filt:
A script file for initialising and running the navigation filter.

set_up:
A script file for initialising an example run for subsequent filtering. It allows graphical input of beacon locations, and vehicle path.

[xest,Pest,innov,S]=update(xpred,Ppred,obs,beacons):
A function to do a one-step update of vehicle location.