A Comment on Assegei's use of Kalman Filter for Clock Synchronization of Wireless Sensor Networks

Faranak Heidarian

April, 2010

1 An Introduction to Kalman Filter

An alternative to the median algorithm is based on Kalman filter[1] which addresses the general problem of trying to estimate the state $x \in R^n$ of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-w} \tag{1}$$

with a measurement $z \in R^m$ that is

$$z_k = Hx_k + v_k. \tag{2}$$

The random variables w_k and v_k represent the process and the measurement noise(respectively). They are assumed to be independent (of each other), white, and with normal probability distributions

$$p(w) \sim N(0, Q),$$

$$p(v) \sim N(0, R).$$

In practice, the *process noise covariance* Q and *measurement noise covariance* R matrices might change with each time step or measurement, however here we assume they are constant,

The $n \times n$ matrix A in the difference equation (1) relates the state of the previous time step $k-1$ to the state at the current step k, in the absence of either a driving function or process noise. The $n \times l$ matrix B relates the optional control input $u \in R^l$ to the state x. The $m \times n$ matrix H in the measurement equation (2) relates the state to the measurement z_k. We assume A, B and H are constant.

The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in form of (noisy) measurements. As such, the the equations for the Kalman filter fall into two groups: *time update* equations and *measurement* equations. The time update equations are responsible for projecting forward (in time) the current sate and error covariance estimates to obtain the *a priori* estimate for the next time step. The measurement update equations are responsible for the feedback–i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate. The time update equations can also be thought of as *predictor* equations, while the measurement update equations can be thought of as *corrector* equations.

We define $\hat{x}_k^- \in R^n$ to be our *a priori* state estimate at step k given knowledge of process prior to step k, and $\hat{x}_k \in R^n$ to be our *a posteriori* state estimate at step k given measurement z_k. We can also define *a priori* and *a posteriori* error covariances P_k^- and P_k. Then, the speific equations for time and measurement updates are presented below in Table 1 and Table 2.

Table 1: Discrete Kalman filter time update equations

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{4}$$

2 Synchronization Algorithm

The implementation of Kalman filter on sensor nodes is shown in linsting 1. The code is used by CHESS on their WSN simulator and is based on the formulas of [2].

Table 2: Discrete Kalman filter time update equations

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (5)$$

$$\hat{x}_k = A\hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \quad (6)$$

$$P_k = (I - K_k H) P_k^- \quad (7)$$

Listing 1: the algorithm of Kalman filter implemented in C in CHESS

```
float kalmanFilter() {
    float x=0;//estimated value of the offset
    float A=1;//state transition matrix
    float int Q=1;//noise covariance of process
    float H = 1;//measurement updating factor
    double R=0.0001;//noise covariance
    float P;//error covariance of measurement
    float K;//Kalman gain
    P=1;//initial estimate of error covariance matrix
    //We assume that nNeighbors is at least 3
    for (int i=0; i<nNeighbors; i++) {
        //time update "PREDICT" equations
        x=A*x;
        P= A*P*A+Q;//Assegei, Eq 4.29
        //measurement update "CORRECT"
        //first step:compute Kalman gain
        if(P+R==0) {
            K=P;
        } else {
            K=P*H/((H*P*H)+R);   //Assegei, Eq 4.34
        }
        //second step:update estimate with measurement
        x=x+K*(phaseError[i]-(H*x));//Assegei, Eq 4.31
        //update the error covariance
        P=(1-K*H)*P;//Assegei, Eq 4.32
    }
    return x;
}
```

It is not easy(possible?) to analyze the algorithm of Kalman filter, using Uppaal or any other formal model checker, as a key parameter of the algorithm, the error covariance, is a real value between 0 and 1, which necessitate floating point computations.

However, we looked deeply into the code of Kalman filter implemented by CHESS for clock synchronization of wireless sensor networks and found a serious bug in it. We noticed that the covariance matrix is initialized every time the function is called (listing 1, line 9), which cancels out the learning ability of Kalman filter.

We a made small program in C++ to simulate the behavior of a network of 4 nodes with clique topology on which Kalman filter is used for synchronization.

Network Size	4
Slot Size	30
Guard Time	6
Tick Length[0]	100000
Tick Length[1]	99999
Tick Length[2]	99999
Tick Length[3]	100000

Table 3: The Specification of a Sample 4-Node WSN running Kalman Filter

For the setting of table 3, the network crashed in the second frame after node 2 could not receive the massage of node 1, because of inappropriate slot number. The error scenario is summarized in table 4.

Sender	Receivers			
	0	1	2	3
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
0	0	0	0	0
1	1	1	0	1

Table 4: An Error Scenario of a 4-Node WSN running Kalman Filter

References

[1] G. Welch and G. Bishop, "An Introduction to Kalman Filter", TR 95-041. Technical report, Department of Computer Science, University of North Carolina at Chapel Hill, 2002.

[2] F. A. Assegei, "Decentralized frame synchronization of a TDMA-based wireless sensor network", Masters thesis, Eindhoven University of Technology, Department of Electrical Engineering, 2008.