Sensor fusion

Wittrock, E.P.

Award date:
1996

Eindhoven University of Technology
Department of Electrical Engineering
Measurement and Control Group

SENSOR FUSION

E.P. Wittrock

M.Sc.Thesis
Carried out from February 1996 to October 1996
Commissioned by Prof. dr. ir. P.P.J. van den Bosch
Under supervision of Dr. S. Weiland and ir. R. Kunst
Eindhoven, 10 October 1996

Summary

Wittrock, E.P.;
<u>Sensor Fusion</u>
M.Sc. Thesis, Measurement and Control Group MBS, section ER, Electrical Engineering, Eindhoven University of Technology, The Netherlands, Oct. 1996.

Sensor fusion is used to combine sensor measurements with different properties to form a resulting measurement that may not be possible from one single sensor alone. Objective of this report is to find methods, and compared them with each other, to fuse several sensor signals. Applications in the section ER are in two projects. The first project needs to measure the deck position of a schip and the second project needs to measure the position of a bus on a bus lane. This report describes discrete time Kalman filter based algorithms used for Sensor Fusion. Several approaches have been proposed and defined. The covariance based algorithms are used to fuse two or more separate sensor measurements and merge them to one entity. The various fusion algorithms are implemented in Matlab files and the performance of the algorithms are evaluated through computer simulations with a one-dimensional dynamic model, with a assumed acceleration acting on the position and speed. It is shown that the defined fusion algorithms give similar results under various sensor configurations and sensor variances. Also investigated is the effect of bias on the various fusion algorithms. A proposal is done to solve the sensor fusion problem with a H_∞ filter, to enable us to take into account the sensor properties with weighting functions in the frequency domain.

Samenvatting

Wittrock, E.P.;
<u>Sensor Fusie</u>
Afstudeerverslag, vakgroep MBS, sectie ER, Faculteit Elektrotechniek, Technische Universiteit Eindhoven, Okt. 1996.

Sensor fusie word gebruikt om verscheidene sensor metingen met verschillende eigenschappen te combineren tot een resultaat dat niet kan worden bereikt met een enkele sensor. Doelstelling van dit raport is methoden te vinden, en met elkaar te vergelijken, om verschillende sensor signalen the fuseren. Toepassingen voor deze vakgroep zijn er voor een tweetal projecten. Het eerste project is de bepaling van een dekstand van een schip en bij het tweede project is het van belang de positie van een bus op een busbaan zeer nauwkeurig te bepalen. Dit rapport beschrijft Kalman filter algoritmen, in de discrete tijd, gebruikt om de sensor metingen te fuseren. Verscheidene methoden zijn voorgesteld en gedefinieerd. De op covariantie gebaseerde algoritmen worden gebruikt om twee of meerdere verschillende sensor metingen te fuseren tot een gezamenlijk resultaat. De verscheidene fusie algoritmen zijn geïmplementeerd in Matlab files en de prestaties van de algoritmen zijn geëvalueerd met behulp van computer simulaties met een 1-dimensionaal dynamisch model, met een versnelling die inwerkt op de snelheid en de positie. De simulaties laten zien dat de algoritmen hetzelfde resultaat geven met verscheidene sensor configuraties en sensor varianties. Ook is er gekeken naar het effect van bias op de verscheidene fusie algoritmen. Er wordt een voorstel gedaan om het sensor fusie probleem op de lossen met een H_∞ filter, op een equivalente manier als het Kalman filter, waardoor we de sensor eigenschappen in het frequentie bereik kunnen meenemen in de weegfuncties.

TABLE OF CONTENTS

List of abbreviations

$\boldsymbol{x}_k$	: state vector at time k, with dim($\boldsymbol{x}_k$)=n
$\boldsymbol{x}_{i_k}$	: local state vector at time k, with dim($\boldsymbol{x}_{i_k}$)= n_i
$\boldsymbol{\Phi}_k$	: known state transition matrix, possible time varying
w_k	: white Gaussian vector noise with variance matrix $\boldsymbol{Q}_k$
$\boldsymbol{\Gamma}_k$	: known process noise transition matrix, possible time varying
z_k^i	: measurement vector of sensor i at time k
z_k	: global measurement vector at time k, by combining z_k^i
v_k^i	: white Gaussian vector noise of sensor i with variance matrix $\boldsymbol{R}_k$
$\boldsymbol{H}^i$	: observation matrix of sensor i
$\hat{\boldsymbol{x}}_{k/k}$	: global state estimate at time k
$\hat{\boldsymbol{x}}_{k/k-}$	: predicted global state estimate at time k
$\boldsymbol{P}_{k/k}$	: global covariance of sensor i at time k
$\boldsymbol{P}_{k/k-1}$	: predicted global covariance of sensor i at time k
$\hat{\boldsymbol{x}}_{i_{k/k}}$	: local state estimate of sensor i at time k
$\hat{\boldsymbol{x}}_{i_{k/k-1}}$	: predicted local state estimate of sensor i at time k
$\boldsymbol{P}_{i_{k/k}}$	: local covariance of sensor i at time k
$\boldsymbol{P}_{i_{k/k-1}}$	: predicted local covariance of sensor i at time k
n	: dimension of the state vector
N	: amount of sensors
$\boldsymbol{R}$	: sensor noise variance matrix
$\boldsymbol{R}^i$	: sensor noise variance vector of sensor i
$\boldsymbol{Q}_k$	: process noise
n_i	: dimension of the state vector of sensor i
$\boldsymbol{T}_i$	: nodal transformation matrix
$\boldsymbol{C}_i$	: local state observation matrix, of the distributed model architecture
$\boldsymbol{\Psi}_k$	: known input transition matrix, possible time varying.
u_k	: system input vector.

Chapter 1. Introduction

Using different types of sensors to obtain information allows the advantages of one sensor type to compensate for the disadvantages of another and further provides redundancy, increasing system robustness.
In this report different kinds of architectures for multi-sensor data fusion will be discussed:
■ In the centralized fusion architecture [chapter 3] all the raw sensor data is transmitted to the fusion agent. The main advantage is that the algorithm is simple and is conceptually similar to single sensor algorithm. A disadvantage of this approach may be the high computational load of the central processor.

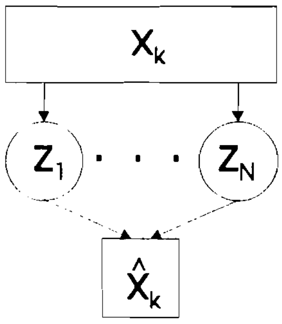

Figure 1, centralized fusion

■ The hierarchical architecture [chapter 4], frequently referred to as the sensor level tracking in which each sensor maintains its own track full based on its own data.

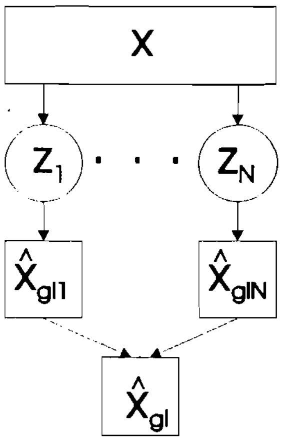

Figure 2, hierarchical fusion

The tracks from the various sensors are transmitted to a single central processor which is responsible for fusing the tracks to form a global estimate. An advantage is that it is easier to detect errors in the sensors by checking the sensor-level tracks with the central tracks. A disadvantage of this approach is the need for two types of algorithms: one for sensor level tracking and the other for data fusion. Another disadvantage is that the system model in the fusion centre is of the same size as in the local filter/local tracker, what leads to a bigger computational load as compared to the hierarchical distributed model architecture.

■ In the hierarchical architecture with feedback [chapter 5] the global state estimate is fed back to the local agents, then each sensor maintains its track based on its own data and the data communicated back from the fusion centre to the local filter. The disadvantages of this method are that there is the need for two types of algorithms (as in the hierarchical method) and the need for extra communication between fusion centre and local filters.

In [8, Alouani-AT] this method is modified so that the communication requirements between the central processor and the local sensors is reduced.

■ In the hierarchical distributed model architecture [chapter 6] the global model is distributed through the different nodes, so that the models at the local nodes are smaller and more appropriate to the dynamics of its observation. The advantage of this method is that because of the reduced order of the system model in the local nodes the computational load is smaller as compared to the hierarchical architecture.

■ In the decentralized architecture the central processing is absent but all nodal processors implement the same model yielding a robust but greatly complicated network requiring full connectedness and excessive computation and communication. In this architecture no central processor is necessary, globally estimates are obtained at each node.

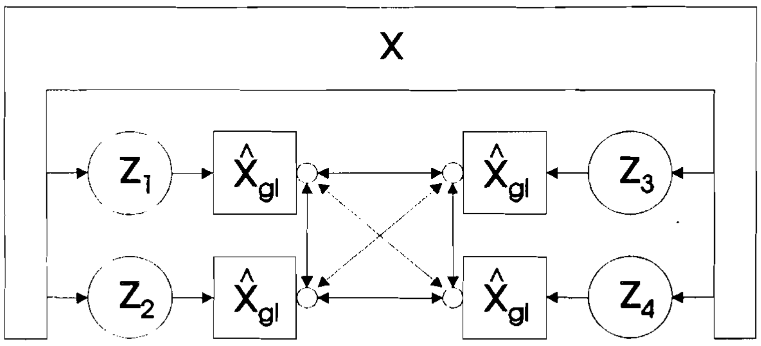

Figure 3, decentralized architecture

■ In the decentralized distributed model architecture the global system model is distributed among a decentralized network. Model distribution reduces the computational burden compared to the decentralized architecture, also the communication between the nodes will be reduced.

In this report both the decentralized architectures are not discussed. Because of their decentralized architecture they offer reduced vulnerability as compared to their centralized or hierarchical counterparts. This reduced vulnerability is of main interest in the military command and control, but no advantage is found for the applications in our faculty. The references to this fusion architectures can be found [10] and more in its bibliography.

Chapter 2. System model

The multi-sensor system is modeled as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{\Phi}_k \boldsymbol{x}_k + \boldsymbol{\Psi}_k u_k + \boldsymbol{\Gamma}_k w_k \qquad (2,1)$$

$\boldsymbol{x}_k$	: state vector at time k, with dim($\boldsymbol{x}_k$)=n.
$\boldsymbol{\Phi}_k$	: known state transition matrix, possibly time varying.
w_k	: white Gaussian vector noise with variance matrix $\boldsymbol{Q}_k$.
$\boldsymbol{\Gamma}_k$	: known process noise transition matrix, possibly time varying.
$\boldsymbol{\Psi}_k$	: known input transition matrix, possibly time varying.
u_k	: system input vector.

The process noise w_k is described by a gaussian random process with zero mean and known variance:

$$E[w_k] = 0$$
$$E[w_k w_l] = \sigma_w^2 \delta_{kl} = \boldsymbol{Q}_k \delta_{kl}$$

The measurement equation of the N sensors is given by:

$$z_k^i = \boldsymbol{H}^i \boldsymbol{x}_k + v_k^i \qquad i = 1, \ldots, N \qquad (2,2)$$

v_k^i	: white Gaussian vector noise with variance matrix $\boldsymbol{R}_k$.
$\boldsymbol{H}^i$	: sensor observation matrix, with dim($\mathbf{H}^i$)=(1*n) .

The noise v_k^i is described by a gaussian random process with zero mean and known variance:

$$E[v_k^i] = 0$$
$$E[v_k^i (v_l^i)^T] = \sigma_{v^i}^2 \delta_{kl} = \boldsymbol{R}_k^i \delta_{kl}$$

The noises are assumed to be uncorrelated.

If there is a bias in the sensor measurement then this can be expressed by non zero mean noise $E[v^i_k] \neq 0$. What the effect of bias is on the sensor fusion will be investigated in the simulations. The question here is that either the bias will be amplified or weakened in the fused sensor signal. We also want to look what happens with an bias in one of the sensor signals.

Chapter 3. Central fusion

In [5, page.250] the central fusion is discussed. In the central fusion approach the sensor measurements are fused to provide the optimal state estimate. Optimal in this case means that the state vector $\mathbf{x}_k$ is reconstructed so that the square mean value (the covariance) of the error will be minimal. The central fusion architecture is shown in Figure 4. As described in [3, page 275] an input vector u_k is added to the Kalman filter. This input vector u_k is only used in the equation for the predicted state estimate (3,6) and because of this there is no effect in the fusion algorithms derived in this report.

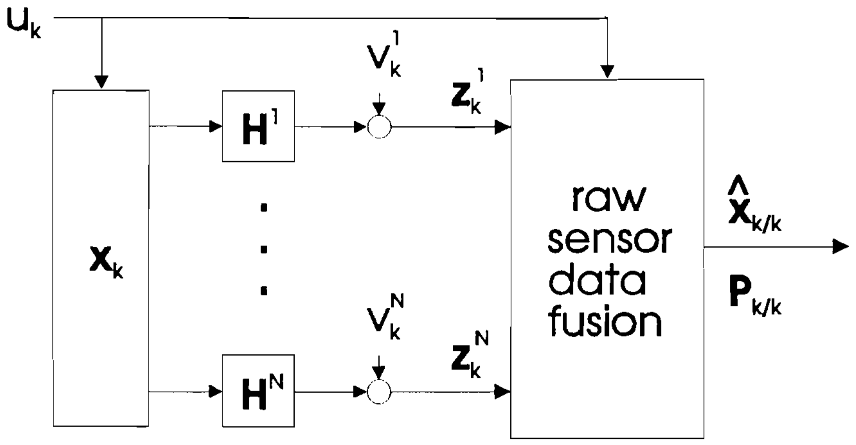

Figure 4, central fusion method

With central fusion all the sensor data is collected by the fusion agent, the global observation would be:

$$z_k \triangleq H x_k + v_k \tag{3,1}$$

Where:

$$\begin{aligned} z_k &\triangleq [(z_k^i)^T, \ldots, (z_k^N)^T]^T \\ H &\triangleq [(H^i)^T, \ldots, (H^N)^T]^T \\ v_k &\triangleq [(v_k^i)^T, \ldots, (v_k^N)^T]^T \end{aligned} \tag{3,2}$$

The noises are independent so the covariance matrix of the noise $\boldsymbol{v}_k$ is given by the diagonal matrix:

$$R \triangleq diag[R^i, \ldots, R^N] \tag{3,3}$$

The global estimate and covariance with all the sensor data is given by:

$$\hat{x}_{k/k}=\hat{x}_{k/k-1}+K_k(z_k-H\hat{x}_{k/k-1}) \\ \text{with} \quad K_k=P_{k/k}H^TR^{-1} \tag{3,4}$$

and:

$$P_{k/k}=(P_{k/k-1}^{-1}+H^TR^{-1}H)^{-1} \tag{3,5}$$

The predicted estimate and the predicted covariance can be computed from the extrapolation equations:

$$\hat{x}_{k/k-1}=\Phi_{k-1}\hat{x}_{k-1/k-1}+\Psi_{k-1}u_{k-1} \tag{3,6}$$

and:

$$P_{k/k-1}=\Phi_{k-1}P_{k-1/k-1}\Phi_{k-1}^T+\Gamma_{k-1}Q_{k-1}\Gamma_{k-1}^T \tag{3,7}$$

The Kalman gain K_k and the covariance matrix $P_{k/k}$ can be written in two ways. In the way used above the Kalman gain $K_k=P_{k/k}H^TR^{-1}$ is calculated with the covariance matrix. So the covariance matrix is calculated first. These equations are derived from the equations below, where the Kalman gain has to be calculated first with the predicted covariance matrix. Subsequently the covariance matrix can be calculated with the calculated Kalman gain.

$$K_k=P_{k/k-1}H^T(HP_{k/k-1}H^T+R)^{-1} \tag{3,8}$$

$$P_{k/k}=(I-K_kH)P_{k/k-1} \tag{3,9}$$

With the equations above the result are two slightly different recursive loops for calculating the global estimate and the global covariance. With simple substitutions it can be seen that the result is the same. The equations for the two slightly different recursive loops are given in this report because in the literature they are both used in the referred literature. The order of calculating is shown in Table I.

Table I, Difference between recursive calculations

method 1	equations 1	method 2	equations 2
$\hat{x}_{k/k-1}$	(3,6)	$\hat{x}_{k/k-1}$	(3,6)
$\mathbf{P}_{k/k-1}$	(3,7)	$\mathbf{P}_{k/k-1}$	(3,7)
$\mathbf{P}_{k/k}$	(3,5)	$\mathbf{K}_k$	(3,8)
$\mathbf{K}_k$	(3,4)	$\mathbf{P}_{k/k}$	(3,9)
$\hat{x}_{k/k}$	(3,4)	$\hat{x}_{k/k}$	(3,4)

3.1. Selection calculation method

Selecting one of the two recursive calculation, method 1 and method 2 can be based on two criterion namely the numerical stability of the equations and the difference in calculation time necessary for one calculation loop.

A disadvantage of method 1 can be the extra inverse of $\mathbf{P}_{k/k-1}$ introduced in equation (3,5). If $\mathbf{P}_{k/k-1}$ becomes very small this inverse can lead to numerical instabilities, and method 2 will have the preference. A reason to choose for method 1 can be a smaller calculation time compared to method 2.

To calculate the difference in calculation time we want to look at the proportion (V) between one calculation loop of method 1 and one calculation loop of method 2. The calculation time exist of a collective part (A) that exists of the calculation made by equations (3,4), (3,6) and (3,7). The additional calculations (B) for method 1 that are given by $\mathbf{K}_k$ in equation (3,4), and equation (3,5) and the additional calculations (C) for method 2 are given by the equations (3,8) and (3,9). A multiplication of (m*n) matrix with a (n*p) matrix is of the order O(m*n*p) and the inverse of a (n*n) matrix is of order O(0.7*n^3). The value 0.7 is estimated with help of Matlab. Addition of matrixes are not taken in account because they are of lower order and faster than multiplications.

$$\begin{aligned} A&=2n^3+2n^2+n+2Nn \\ B&=2N^2n+2Nn^2+1.4n^3 \\ C&=3Nn^2+2N^2n+0.7N^3+n^3 \\ V &\triangleq \frac{T_{method2}}{T_{method1}}=\frac{A+C}{A+B} \end{aligned} \qquad (3,10)$$

The proportion V is defined by the time necessary for method 2 divided by the time necessary for method 1. With the proportion V we can easily conclude whether method 1 or method 2

is faster, as a function of system order n and amount of sensors N.
With the matlab file in Appendix A the proportion V as a function of system order n is calculated for various amount of sensors N, the result of this can be seen in Figure 5.

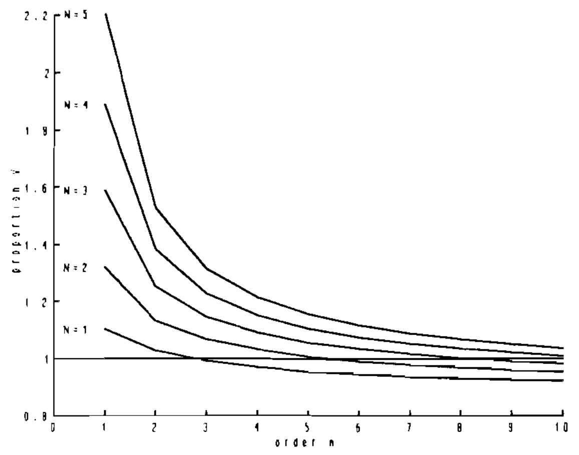

Figure 5, Calculation time proportion method 1 / method 2

If V>1 method 1 is faster and if V<1 method 2 is faster. From the figure can be seen that for growing amount of sensors (N) method 1 is computational more efficient.

So in case there is a risk for numerical instability method 2 has a preference. Risk for numerical instability is can be caused in the case that the process noise variance Q is equal to zero. In that case the covariance matrix $\mathbf{P}_{k/k}$ will go to zero and so the inverse of $\mathbf{P}_{k/k-1}$ will lead to numerical instability. In the case that there is no risk for numerical instability the calculation time is the motive for selecting a method. If the proprotion V is bigger than 1 we prefer method 1 and if the proportion V is smaller than 1 we prefer method 2. In Figure 5 we see that if we use a lot sensors and the order of the system is limited than V>1 and so method 1 is prefered.

Chapter 4. Hierarchical fusion

In [9, Sun-H] an algorithm for the hierarchical fusion is proposed. The new fusion algorithm is obtained by further derivation of the centralized fusion algorithm. First the algorithm will be presented and discussed, subsequently the algorithm will be derived and also compared with the central fusion algorithm.

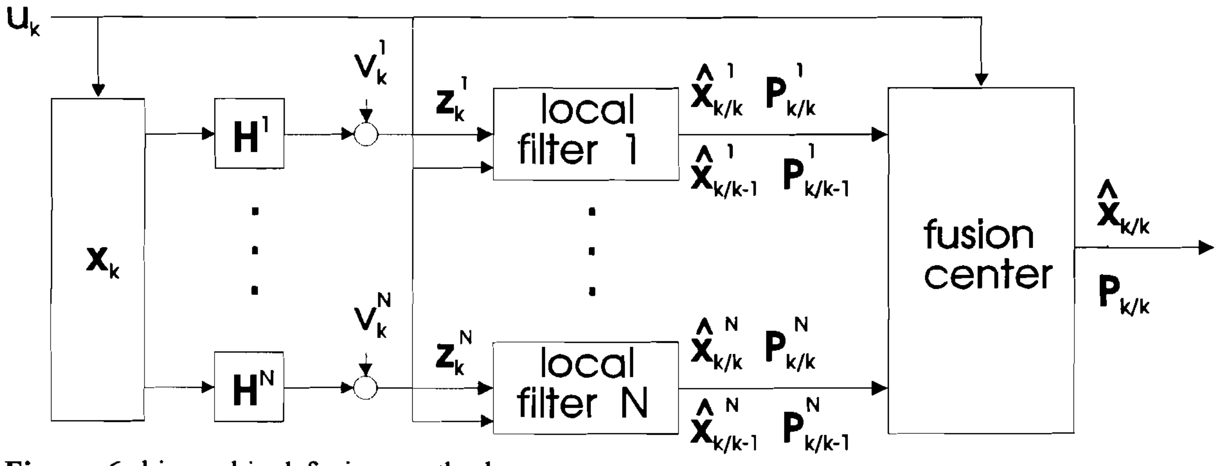
u_k
x_k
H^1
H^N
v_k^1
v_k^N
z_k^1
z_k^N
local filter 1
local filter N
$\hat{x}_{k/k}^1$ $P_{k/k}^1$
$\hat{x}_{k/k-1}^1$ $P_{k/k-1}^1$
$\hat{x}_{k/k}^N$ $P_{k/k}^N$
$\hat{x}_{k/k-1}^N$ $P_{k/k-1}^N$
fusion center
$\hat{x}_{k/k}$
$P_{k/k}$

Figure 6, hierarchical fusion method

The hierarchical fusion method needs two types of processing nodes:

■ Local tracking node. The purpose of the local tracking node is to obtain a local state estimate and a local covariance. The local state estimates are all of the same order as the global state estimate, namely order n. A way of reducing this order is discussed in chapter 6.

■ Data fusion centre. This centre fuses the local estimates to a global estimate, with help of the local covariances.

In the local node the filter processes data from a local sensor to generate a local state estimation. At each time k the local agents transmit the local state estimate and the predicted local state estimate, as well as the corresponding covariances to the fusion agent. It is not necessary to communicate the predicted local state estimate and predicted local covariance to the fusion centre because they can also be calculated in the fusion centre with the local state estimate and the local covariance. If the predicted local state estimate and the predicted local covariance are calculated in the fusion center you make a trade between extra calculation at the fusion center and less communication between local filter and the fusion center.
In the fusion centre, the local state estimates are fused with the predicted global state estimate to produce a global state estimate. This fusion is weighted with the local covariances and the global covariance. The predicted global state estimate and the predicted global covariance are obtained with the extrapolation equations (3,6) and (3,7). To calculate the predicted global state estimate you need to connect the input u_k to the fusion center, this is caused by the need for u_k in equation (3,6). .

According to [5, page 255] the data fusion equations in the data fusion centre can be written as:

$$P_{k/k}^{-1}\hat{x}_{k/k}=P_{k/k-1}^{-1}\hat{x}_{k/k-1}+\sum_{i=1}^{N}\left((P_{k/k}^{i})^{-1}\hat{x}_{k/k}^{i}-(P_{k/k-1}^{i})^{-1}\hat{x}_{k/k-1}^{i}\right) \qquad (4,1)$$

$$P_{k/k}^{-1}=P_{k/k-1}^{-1}+\sum_{i=1}^{N}\left((P_{k/k}^{i})^{-1}-(P_{k/k-1}^{i})^{-1}\right) \qquad (4,2)$$

Further derivation [9, Sun-H] the data fusion equations will give the following equations for the global state estimate:

$$\hat{x}_{k/k}=\hat{x}_{k/k-1}+P_{k/k}\left(\sum_{i=1}^{N}(P_{k/k}^{i})^{-1}(\hat{x}_{k/k}^{i}-\hat{x}_{k/k-1})+\sum_{i=1}^{N}(P_{k/k-1}^{i})^{-1}(\hat{x}_{k/k-1}-\hat{x}_{k/k-1}^{i})\right) \qquad (4,3)$$

This equation expresses that the global state estimate equals the sum of the global state prediction and an update. This update involves two terms. One is the sum of the differences between each local state estimate and fusion prediction, weighted by the inverse of the corresponding local sensor covariance. The other one is the sum of the differences between the fusion prediction and each local sensor prediction, weighted by the inverse of the corresponding local sensor prediction covariance. The sum of the two terms is weighted by fusion state covariance.

The global covariance is calculated according to equation (4,2) , and the predicted global estimate and the predicted global covariance are calculated with the extrapolation equations (3,6) and (3,7).

4.1. Derivation of the hierarchical fusion algorithm

The purpose of the fusion centre is to calculate the global estimate and the global covariance with help of the local estimates, local covariances, predicted global estimate and predicted global covariance. The hierarchical fusion algorithm can be derived by substitutions of the local estimator equations in the central fusion equations.

The local estimate and covariance are calculated with:

$$\hat{x}_{k/k}^{i}=\hat{x}_{k/k-1}^{i}+P_{k/k}^{i}H^{i^{T}}(R^{i})^{-1}\,(z_{k}^{i}-H^{i}\hat{x}_{k/k-1}^{i}) \qquad (4,4)$$

$$P_{k/k}^{i}=((P_{k/k-1}^{i})^{-1}+H^{i^{T}}(R^{i})^{-1}H^{i})^{-1} \qquad (4,5)$$

The global estimate is calculated with equation (3,4) and the global covariance is calculated with equation (3,5).

The equation (3,2) can be written in the following form.

$$z_k \triangleq \begin{bmatrix} z_k^1 \\ \vdots \\ z_k^N \end{bmatrix} \quad H \triangleq \begin{bmatrix} H^1 \\ \vdots \\ H^N \end{bmatrix} \quad v_k \triangleq \begin{bmatrix} v_k^i \\ \vdots \\ v_k^N \end{bmatrix} \quad R \triangleq \begin{bmatrix} R^1 & 0 & \cdots & 0 \\ 0 & R^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R^N \end{bmatrix} \tag{4,6}$$

With (4,6) it can easily be shown that:

$$H^T R^{-1} H = \sum_{i=1}^{N} (H^{i^T} (R^i)^{-1} H^i)$$

Rewriting (4,5) and combining this result with the equation above will give:

$$\begin{aligned} (P^i_{k/k})^{-1} &= (P^i_{k/k-1})^{-1} + H^{i^T} (R^i)^{-1} H^i \\ H^{i^T} (R^i)^{-1} H^i &= (P^i_{k/k})^{-1} - (P^i_{k/k-1})^{-1} \\ H^T R^{-1} H &= \sum_{i=1}^{N} (H^{i^T} (R^i)^{-1} H^i) \\ &= \sum_{i=1}^{N} ((P^i_{k/k})^{-1} - (P^i_{k/k-1})^{-1}) \end{aligned}$$

Filling in this result in equation (3,5) will give equation (4,2), where the global covariance is calculated with the predicted global covariance, the local covariances and the predicted local covariances.

With (4,6) it can easily be shown that:

$$H^T R^{-1} z_k = \sum_{i=1}^{N} (H^{i^T} (R^i)^{-1} z^i_k)$$

Rewriting equation (4,4) with help of (4,5) and combining this result with the equation above will give:

$$\begin{aligned} &\hat{x}^i_{k/k} = \hat{x}^i_{k/k-1} + P^i_{k/k} H^{i^T} (R^i)^{-1} (z^i_k - H^i \hat{x}^i_{k/k-1}) \\ &(P^i_{k/k})^{-1} (\hat{x}^i_{k/k} - \hat{x}^i_{k/k-1}) + H^{i^T} (R^i)^{-1} H^i \hat{x}^i_{k/k-1} = H^{i^T} (R^i)^{-1} z^i_k \\ &(P^i_{k/k})^{-1} \hat{x}^i_{k/k} + (H^{i^T} (R^i)^{-1} H^i - (P^i_{k/k})^{-1}) \hat{x}^i_{k/k-1} = H^{i^T} (R^i)^{-1} z^i_k \\ &(P^i_{k/k})^{-1} \hat{x}^i_{k/k} - (P^i_{k/k-1})^{-1} \hat{x}^i_{k/k-1} = H^{i^T} (R^i)^{-1} z^i_k \\ &H^T R^{-1} z_k = \sum_{i=1}^{N} ((P^i_{k/k})^{-1} \hat{x}^i_{k/k} - (P^i_{k/k-1})^{-1} \hat{x}^i_{k/k-1}) \end{aligned}$$

Now we will rewrite equation (3,4) so that we can fill in the equation above. The result is equation (4,1).

$$\begin{aligned}
\hat{\boldsymbol{x}}_{k/k} &= \hat{\boldsymbol{x}}_{k/k-1} + \boldsymbol{P}_{k/k}\boldsymbol{H}^T\boldsymbol{R}^{-1}(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k/k-1}) \\
\boldsymbol{P}_{k/k}^{-1}\hat{\boldsymbol{x}}_{k/k} &= \boldsymbol{P}_{k/k}^{-1}\hat{\boldsymbol{x}}_{k/k-1} + \boldsymbol{H}^T\boldsymbol{R}^{-1}(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k/k-1}) \\
&= (\boldsymbol{P}_{k/k-1}^{-1} + \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H})\,\hat{\boldsymbol{x}}_{k/k-1} + \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{z}_k - \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{H}\hat{\boldsymbol{x}}_{k/k-1} \\
&= \boldsymbol{P}_{k/k-1}^{-1}\hat{\boldsymbol{x}}_{k/k-1} + \boldsymbol{H}^T\boldsymbol{R}^{-1}\boldsymbol{z}_k \\
&= \boldsymbol{P}_{k/k-1}^{-1}\hat{\boldsymbol{x}}_{k/k-1} + \sum_{i=1}^{N}\left((\boldsymbol{P}_{k/k}^{i})^{-1}\hat{\boldsymbol{x}}_{k/k}^{i} - (\boldsymbol{P}_{k/k-1}^{i})^{-1}\hat{\boldsymbol{x}}_{k/k-1}^{i}\right)
\end{aligned}$$

With help of equations (4,1) and (4,2) and a more convenient equation for the global state estimate is derived:

$$\boldsymbol{P}_{k/k}^{-1} = \boldsymbol{P}_{k/k-1}^{-1} + \sum_{i=1}^{N}\left((\boldsymbol{P}_{k/k}^{i})^{-1} - (\boldsymbol{P}_{k/k-1}^{i})^{-1}\right)$$

$$\boldsymbol{P}_{k/k}^{-1}\hat{\boldsymbol{x}}_{k/k-1} = \boldsymbol{P}_{k/k-1}^{-1}\hat{\boldsymbol{x}}_{k/k-1} + \sum_{i=1}^{N}\left((\boldsymbol{P}_{k/k}^{i})^{-1} - (\boldsymbol{P}_{k/k-1}^{i})^{-1}\right)\hat{\boldsymbol{x}}_{k/k-1}$$

$$\boldsymbol{P}_{k/k-1}^{-1}\hat{\boldsymbol{x}}_{k/k-1} = \boldsymbol{P}_{k/k}^{-1}\hat{\boldsymbol{x}}_{k/k} - \sum_{i=1}^{N}\left((\boldsymbol{P}_{k/k}^{i})^{-1}\hat{\boldsymbol{x}}_{k/k}^{i} - (\boldsymbol{P}_{k/k-1}^{i})^{-1}\hat{\boldsymbol{x}}_{k/k-1}^{i}\right)$$

$$\begin{aligned}
\boldsymbol{P}_{k/k}^{-1}\hat{\boldsymbol{x}}_{k/k} = \boldsymbol{P}_{k/k}^{-1}\hat{\boldsymbol{x}}_{k/k-1} &+ \sum_{i=1}^{N}\left((\boldsymbol{P}_{k/k}^{i})^{-1}\hat{\boldsymbol{x}}_{k/k}^{i} - (\boldsymbol{P}_{k/k-1}^{i})^{-1}\hat{\boldsymbol{x}}_{k/k-1}^{i}\right) + \\
&- \sum_{i=1}^{N}\left((\boldsymbol{P}_{k/k}^{i})^{-1}\hat{\boldsymbol{x}}_{k/k-1} - (\boldsymbol{P}_{k/k-1}^{i})^{-1}\hat{\boldsymbol{x}}_{k/k-1}\right)
\end{aligned}$$

$$\hat{\boldsymbol{x}}_{k/k} = \hat{\boldsymbol{x}}_{k/k-1} + \boldsymbol{P}_{k/k}\sum_{i=1}^{N}\left((\boldsymbol{P}_{k/k}^{i})^{-1}(\hat{\boldsymbol{x}}_{k/k}^{i} - \hat{\boldsymbol{x}}_{k/k-1})\right) + \boldsymbol{P}_{k/k}\sum_{i=1}^{N}\left((\boldsymbol{P}_{k/k-1}^{i})^{-1}(\hat{\boldsymbol{x}}_{k/k-1} - \hat{\boldsymbol{x}}_{k/k-1}^{i})\right)$$

The result of this derivation is equation (4,3), and so we showed how the fusion equations of chapter 4 are derived.

4.2. Comparing the hierarchical fusion with the central fusion

In this section the global state estimate and the global covariance of the central fusion algorithm will be compared with the global state estimate and the global covariance of the hierarchical fusion algorithm in the two sensor case with equal observation matrixes and equal sensor noise variances. The assumptions done for comparing the central fusion with the hierarchical fusion are that we use two similar sensors, consequently: N=2, $\mathbf{H}_1=\mathbf{H}_2$, $\mathbf{R}_1=\mathbf{R}_2$. The comparison in this section is done for a very special case. If we compare other cases like, more than two sensors (n>2), different observation matrices $\mathbf{H}_i\neq\mathbf{H}_j$ or different sensor variances $\mathbf{R}_i\neq\mathbf{R}_j$ the equations would be to extensive to be compared as in this section. So the other cases are examened with simulations further on in this report.
With the assumptions done we will show that the global state estimate of the central fusion algorithm is equal to the global state estimate of the hierarchical fusion algorithm, as well that the global covariance of the central fusion algorithm is equal to the global covariance of the hierarchical fusion algorithm:

$$\hat{x}^{central}_{k/k} \overset{?}{=} \hat{x}^{hier.}_{k/k}$$

$$P^{central}_{k/k} \overset{?}{=} P^{hier.}_{k/k}$$

First we will calculate the global covariance with the central fusion equation (3,5).

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + H^T R^{-1} H$$

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + \begin{bmatrix} H_1^T & H_2^T \end{bmatrix} \begin{bmatrix} R_1^{-1} & 0 \\ 0 & R_2^{-1} \end{bmatrix} \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}$$

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + H_1^T R_1^{-1} H_1 + H_2^T R_2^{-1} H_2$$

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + 2\, H_1^T R_1^{-1} H_1$$

Then the global covariance is calculated with the hierarchical fusion equation (4,2). As expected the result is the same as calculated before with the central fusion equation.

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + \sum_{i=1}^{2} \left((P^i_{k/k})^{-1} - (P^i_{k/k-1})^{-1} \right)$$

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + \sum_{i=1}^{2} \left(H^{i\,T} (R^i)^{-1} H^i \right)$$

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + 2\, H^{1\,T} (R^1)^{-1} H^1$$

If the global state estimate is calculated with the central fusion equation (3,4) in the two similar sensor case the result will be as follows:

$$\hat{\mathbf{x}}_{k/k}=\hat{\mathbf{x}}_{k/k-1}+\mathbf{P}_{k/k}\mathbf{H}^T\mathbf{R}^{-1}(\mathbf{z}_k-\mathbf{H}\hat{\mathbf{x}}_{k/k-1})$$

$$\hat{\mathbf{x}}_{k/k}=\hat{\mathbf{x}}_{k/k-1}+\mathbf{P}_{k/k}\,[\mathbf{H}^{1^T}\ \mathbf{H}^{1^T}]\begin{bmatrix}(\mathbf{R}^1)^{-1} & 0\\ 0 & (\mathbf{R}^1)^{-1}\end{bmatrix}\left(\begin{bmatrix}z_k^1\\ z_k^2\end{bmatrix}_k-\begin{bmatrix}\mathbf{H}^1\\ \mathbf{H}^1\end{bmatrix}\hat{\mathbf{x}}_{k/k-1}\right)$$

$$\hat{\mathbf{x}}_{k/k}=\hat{\mathbf{x}}_{k/k-1}+\mathbf{P}_{k/k}\,(\mathbf{H}^{1^T}(\mathbf{R}^1)^{-1}(z_k^1+z_k^2)-2\mathbf{H}^{1^T}(\mathbf{R}^1)^{-1}\mathbf{H}^1\hat{\mathbf{x}}_{k/k-1})$$

With equations (4,4) and (4,5) we obtain the following expressions which will be combined with the equation for the global state estimate of the hierarchical fusion algorithm.

$$(\mathbf{P}^i_{k/k})^{-1}\hat{\mathbf{x}}^i_{k/k}-(\mathbf{P}^i_{k/k-1})^{-1}\hat{\mathbf{x}}^i{}_{k/k-1}=\mathbf{H}^{i^T}(\mathbf{R}^i)^{-1}z_k^i$$
$$(\mathbf{P}^i_{k/k-1})^{-1}-(\mathbf{P}^i_{k/k})^{-1}=-\mathbf{H}^{i^T}(\mathbf{R}^i)^{-1}\mathbf{H}^i$$

Then the global state estimate is calculated with the hierarchical fusion equation (4,3)

$$\hat{\mathbf{x}}_{k/k}=\hat{\mathbf{x}}_{k/k-1}+\mathbf{P}_{k/k}\left(\sum_{i=1}^{2}(\mathbf{P}^i_{k/k})^{-1}(\hat{\mathbf{x}}^i_{k/k}-\hat{\mathbf{x}}_{k/k-1})+\sum_{i=1}^{2}(\mathbf{P}^i_{k/k-1})^{-1}(\hat{\mathbf{x}}_{k/k-1}-\hat{\mathbf{x}}^i_{k/k-1})\right)$$

$$\hat{\mathbf{x}}_{k/k}=\hat{\mathbf{x}}_{k/k-1}+\mathbf{P}_{k/k}\sum_{i=1}^{2}(\mathbf{P}^i_{k/k})^{-1}\hat{\mathbf{x}}^i_{k/k}-(\mathbf{P}^i_{k/k-1})^{-1}\hat{\mathbf{x}}^i_{k/k-1}+((\mathbf{P}^i_{k/k-1})^{-1}-(\mathbf{P}^i_{k/k})^{-1})\,\hat{\mathbf{x}}_{k/k-1})$$

$$\hat{\mathbf{x}}_{k/k}=\hat{\mathbf{x}}_{k/k-1}+\mathbf{P}_{k/k}\sum_{i=1}^{2}(\mathbf{H}^{i^T}(\mathbf{R}^i)^{-1}z_k^i-\mathbf{H}^{i^T}(\mathbf{R}^i)^{-1}\mathbf{H}^i\,\hat{\mathbf{x}}_{k/k-1})$$

$$\hat{\mathbf{x}}_{k/k}=\hat{\mathbf{x}}_{k/k-1}+\mathbf{P}_{k/k}\,(\mathbf{H}^{1^T}(\mathbf{R}^1)^{-1}(z_k^1+z_k^2)-2\mathbf{H}^{1^T}(\mathbf{R}^1)^{-1}\mathbf{H}^1\,\hat{\mathbf{x}}_{k/k-1})$$

It can be seen that the result of the global state estimate with the central fusion algorithm is corresponding with the result with the hierarchical fusion algorithm.

Chapter 5. Hierarchical fusion with feedback

In [5, page 252] the hierarchical fusion with feedback is presented. The local estimates are communicated to a central location where data fusion is performed, subsequently the fused data is communicated back to the local agents where it is used as priori statistics.

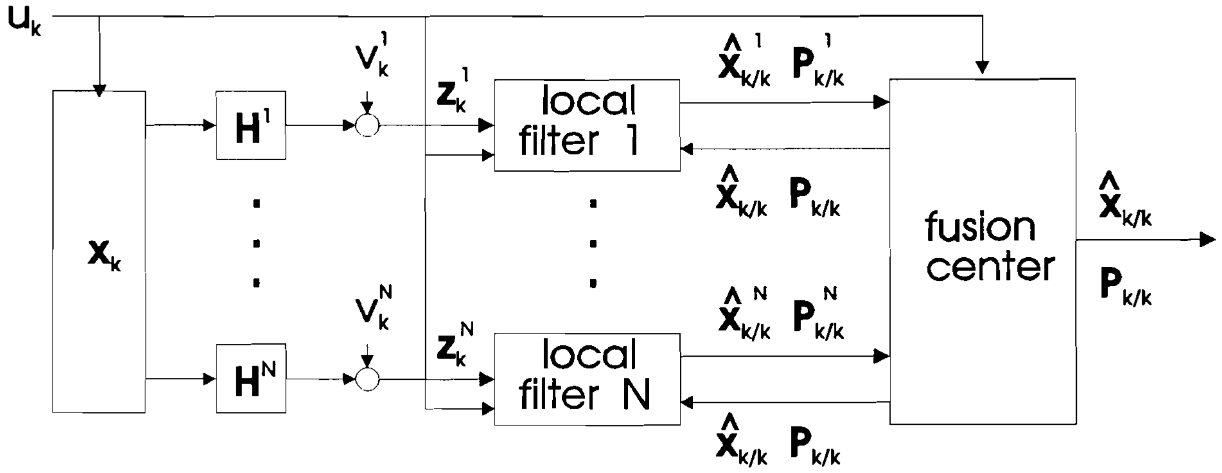

Figure 7, hierarchical fusion with full feedback

The local state estimate and local covariance are calculated in the local filters with the same equations as used in the hierarchical fusion method, namely equation (4,4) and equation (4,5). The difference with the hierarchical fusion method is that in this case the local filters use the global state estimate of the fusion center to calculate the predicted local state estimate.

The fusion equations with feedback are:

$$P^{-1}_{k/k}\hat{x}_{k/k}=\sum_{i=1}^{N}((P^{i}_{k/k})^{-1}\hat{x}^{i}_{k/k})-(N-1)P^{-1}_{k/k-1}\hat{x}_{k/k-1} \tag{5,1}$$

$$P^{-1}_{k/k}=\sum_{i=1}^{N}((P^{i}_{k/k})^{-1})-(N-1)P^{-1}_{k/k-1} \tag{5,2}$$

A more convenient way of writing the equation for the global state estimate is given in equation (5,3).

$$\hat{x}_{k/k}=\hat{x}_{k/k-1}+P_{k/k}\sum_{i=1}^{N}((P^{i}_{k/k})^{-1}(\hat{x}^{i}_{k/k}-\hat{x}_{k/k-1})) \tag{5,3}$$

The predicted global estimate and the predicted global covariance can be computed from the extrapolation equations (3,6) and (3,7). Because of equations (3,6) the input u_k must be connected to the fusion center.

5.1. Derivation of the hierarchical fusion with feedback algorithm

After the global state estimate and the global covariance are calculated they are communicated to the local agents where they are used to predict the local state estimates and the local covariances. The result of this communication can be seen in equation (5,4) and (5,5).

$$P^i_{k-1/k-1} = P_{k-1/k-1} \quad \Rightarrow \quad P^i_{k/k-1} = P_{k/k-1} \tag{5,4}$$

$$\hat{x}^i_{k-1/k-1} = \hat{x}_{k-1/k-1} \quad \Rightarrow \quad \hat{x}^i_{k/k-1} = \hat{x}_{k/k-1} \tag{5,5}$$

When we substitute the predicted local covariances by the predicted global covariance (5,4), the fusion algorithm for the global covariance (4,2) can be rewritten as equation (5,2)

$$\begin{aligned} P^{-1}_{k/k} &= P^{-1}_{k/k-1} + \sum_{i=1}^{N} \left((P^i_{k/k})^{-1} - (P^i_{k/k-1})^{-1} \right) \\ &= P^{-1}_{k/k-1} + \sum_{i=1}^{N} \left((P^i_{k/k})^{-1} - P^{-1}_{k/k-1} \right) \\ &= \sum_{i=1}^{N} \left((P^i_{k/k})^{-1} \right) - (N-1)\, P^{-1}_{k/k-1} \end{aligned}$$

Substituting the predicted local state estimate by the predicted global state estimate (5,5) and substituting the predicted local covariances by the predicted global covariance (5,4), the algorithm for the global state estimate (4,1) can be rewritten as (5,1).

$$\begin{aligned} P^{-1}_{k/k} \hat{x}_{k/k} &= P^{-1}_{k/k-1} \hat{x}_{k/k-1} + \sum_{i=1}^{N} \left((P^i_{k/k})^{-1} \hat{x}^i_{k/k} - (P^i_{k/k-1})^{-1} \hat{x}^i_{k/k-1} \right) \\ &= P^{-1}_{k/k-1} \hat{x}_{k/k-1} + \sum_{i=1}^{N} \left((P^i_{k/k})^{-1} \hat{x}^i_{k/k} - P^{-1}_{k/k-1} \hat{x}_{k/k-1} \right) \\ &= \sum_{i=1}^{N} \left((P^i_{k/k})^{-1} \hat{x}^i_{k/k} \right) - (N-1)\, P^{-1}_{k/k-1} \hat{x}_{k/k-1} \end{aligned}$$

To get a more convenient equation for the global state estimate than equation (5,1) the feedback equation are substituted in equation (4,3).

$$\begin{aligned} \hat{x}_{k/k} &= \hat{x}_{k/k-1} + P_{k/k} \left(\sum_{i=1}^{N} (P^i_{k/k})^{-1} (\hat{x}^i_{k/k} - \hat{x}_{k/k-1}) + \sum_{i=1}^{N} (P^i_{k/k-1})^{-1} (\hat{x}_{k/k-1} - \hat{x}^i_{k/k-1}) \right) \\ &= \hat{x}_{k/k-1} + P_{k/k} \left(\sum_{i=1}^{N} (P^i_{k/k})^{-1} (\hat{x}^i_{k/k} - \hat{x}_{k/k-1}) + \sum_{i=1}^{N} P^{-1}_{k/k-1} (\hat{x}_{k/k-1} - \hat{x}_{k/k-1}) \right) \\ &= \hat{x}_{k/k-1} + P_{k/k} \sum_{i=1}^{N} (P^i_{k/k})^{-1} (\hat{x}^i_{k/k} - \hat{x}_{k/k-1}) \end{aligned}$$

The result of this derivation is equation (5,3).

5.2. Comparing the hierarchical feedback fusion with the central fusion

As seen in section 4.2 the hierarchical feedback fusion algorithm is compared with the central fusion algorithm in the two similar sensor case. The calculated global state estimate and the global covariance with the central fusion algorithm for the two similar sensor case are calculated in section 4.2 and are as follows:

$$\begin{aligned}
&\hat{x}_{k/k}=\hat{x}_{k/k-1}+P_{k/k}\,(H^{1^T}(R^1)^{-1}(z_k^1+z_k^2)-2H^{1^T}(R^1)^{-1}H^1\hat{x}_{k/k-1})\\
&P_{k/k}^{-1}=P_{k/k-1}^{-1}+2H^{1^T}(R^1)^{-1}H^1
\end{aligned}$$

First we will calculate the global covariance with the hierarchical feedback equation (5,2). As expected the result is the same as in the central case.

$$\begin{aligned}
P_{k/k}^{-1}&=\sum_{i=1}^{N}((P_{k/k}^i)^{-1})-(N-1)P_{k/k-1}^{-1}\\
&=(P_{k/k}^i)^{-1}+(P_{k/k}^2)^{-1}-P_{k/k-1}^{-1}\\
&=P_{k/k-1}^{-1}+((P_{k/k}^1)^{-1}-P_{k/k-1}^{-1})+((P_{k/k}^2)^{-1}-P_{k/k-1}^{-1})\\
&=P_{k/k-1}^{-1}+((P_{k/k}^1)^{-1}-(P_{k/k-1}^1)^{-1})+((P_{k/k}^2)^{-1}-(P_{k/k-1}^2)^{-1})\\
&=P_{k/k-1}^{-1}+H^{1^T}(R^1)^{-1}H^1+H^{2^T}(R^2)^{-1}H^2 \qquad .\\
&=P_{k/k-1}^{-1}+2H_1^TR_1^{-1}H_1
\end{aligned}$$

We also calculate the global state estimate with the hierarchical feedback equation (5,3). This result is also the same as in the central case.

$$\begin{aligned}
&\hat{x}_{k/k}^i-\hat{x}_{k/k-1}^i=P_{k/k}^iH^{i^T}(R^i)^{-1}\,(z_k^i-H^i\hat{x}_{k/k-1}^i)\\
&(P_{k/k}^i)^{-1}(\hat{x}_{k/k}^i-\hat{x}_{k/k-1}^i)=H^{i^T}(R^i)^{-1}\,(z_k^i-H^i\hat{x}_{k/k-1}^i)\\
&(P_{k/k}^i)^{-1}(\hat{x}_{k/k}^i-\hat{x}_{k/k-1})=H^{i^T}(R^i)^{-1}\,(z_k^i-H^i\hat{x}_{k/k-1})
\end{aligned}$$

$$\begin{aligned}
\hat{x}_{k/k}&=\hat{x}_{k/k-1}+P_{k/k}\sum_{i=1}^{2}((P_{k/k}^i)^{-1}(\hat{x}_{k/k}^i-\hat{x}_{k/k-1}))\\
&=\hat{x}_{k/k-1}+P_{k/k}\sum_{i=1}^{2}(H^{i^T}(R^i)^{-1}\,(z_k^i-H^i\hat{x}_{k/k-1}))\\
&=\hat{x}_{k/k-1}+P_{k/k}\,(H^{1^T}(R^1)^{-1}\,(z_k^1-H^1\hat{x}_{k/k-1})+H^{2^T}(R^2)^{-1}\,(z_k^2-H^2\hat{x}_{k/k-1}))\\
&=\hat{x}_{k/k-1}+P_{k/k}\,(H^{1^T}(R^1)^{-1}\,(z_k^1+z_k^2)-2H^{1^T}(R^1)^{-1}H^1\hat{x}_{k/k-1}))
\end{aligned}$$

The result of the hierarchical feedback fusion the same as the central fusion in the two similar sensor case.

5.3. Reduced communication requirements

In [8, Alouani-AT] the communication requirements between central processor and local sensors is reduced. In this paper the case of two sensors is considered. In the conclusions is stated that this method can be extended to an arbitrary number of sensors. But nothing is stated, for the case with more than 2 sensors, about the expected communication requirements. Possibly only communications will be required to the first local filter and no communication to the rest of the local filters or maybe only no communication will be required to the last local filter (local filter N) and communication to the rest of the local filters, is still necessary.

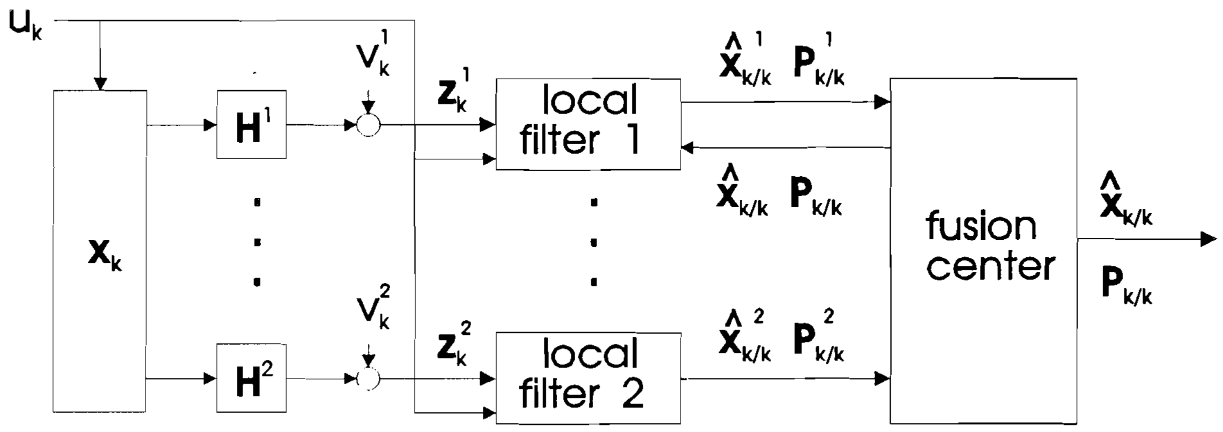

Figure 8, two sensor hierarchical fusion with reduced communication requirements

Special equations for the fusion center are derived in the two sensor case, so that if the fused global state estimate and the fused global covariance is communicated to local filter 1, which uses this information as priori statistics, and no communication is required to local filter 2, the result will be the same as with the hierarchical fusion with full feedback. The gain of this method is that there is no communication from the fusion centre to local filter 2, with no degradation of performance, what is possible by the special derived equations for the fusion center.

The state estimate and its error covariance of sensor 2 is given by equations (5,6) and (5,7).

$$\hat{x}^2_{k|k} = \hat{x}^2_{k|k-1} + P^2_{k|k} H^{2^T} (R^2)^{-1} (z^2_k - H^2 \hat{x}^2_{k|k-1}) \qquad (5,6)$$

$$P^2_{k|k} = ((P^2_{k|k-1})^{-1} + H^{2^T} (R^2)^{-1} H^2)^{-1} \qquad (5,7)$$

The predicted state estimate and the predicted covariance of sensor 2 are calculated with equation (5,8).

$$\begin{aligned} \hat{x}^2_{k|k-1} &= \Phi_{k-1} \hat{x}^2_{k-1|k-1} + \Psi_{k-1} u_{k-1} \\ P^2_{k|k-1} &= \Phi_{k-1} P^2_{k-1|k-1} \Phi^T_{k-1} + \Gamma_{k-1} Q_{k-1} \Gamma^T_{k-1} \end{aligned} \qquad (5,8)$$

The state estimate and its error covariance of sensor 1 is given by equation (5,9) and equation (5,10).

$$\hat{x}^1_{k/k} = \hat{x}^1_{k/k-1} + P^1_{k/k} H^{1^T} (R^1)^{-1} (z^1_k - H^1 \hat{x}^1_{k/k-1}) \qquad (5,9)$$

$$P^1_{k/k} = ((P^1_{k/k-1})^{-1} + H^{1^T} (R^1)^{-1} H^1)^{-1} \qquad (5,10)$$

The predicted state estimate and the predicted covariance of sensor 1 are calculated with equation (5,11).

$$\begin{aligned} \hat{x}^1_{k/k-1} &= \Phi_{k-1} \hat{x}_{k-1/k-1} + \Psi_{k-1} u_{k-1} \\ P^1_{k/k-1} &= \Phi_{k-1} P_{k-1/k-l} \Phi^T_{k-1} + \Gamma_{k-1} Q_{k-1} \Gamma^T_{k-1} \end{aligned} \qquad (5,11)$$

Note that in equation (5,11) the predicted state estimate and the predicted covariance are calculated with help of the previous global state estimate and global covariance of the fusion algorithm. This is because of the feedback of the fused global state estimate and the fused global covariance.

The estimates and covariances of sensor 1 and sensor 2 are now communicated to the central fusion agent where they are fused to a global state estimate and global covariance, with the fusion algorithm for the two sensor case:

$$\hat{x}_{k/k} = P_{k/k}((P^1_{k/k})^{-1} \hat{x}^1_{k/k} + (P^2_{k/k})^{-1} \hat{x}^2_{k/k} - ((P^2_{k/k})^{-1} - H^{2^T} (R^2)^{-1} H^2) \Phi_{k-1} \hat{x}^2_{k-1/k-1}) \qquad (5,12)$$

$$P_{k/k} = (I - K^f_k H^2) P^1_{k/k} \qquad (5,13)$$

$$K^f_k = P^1_{k/k} H^{2^T} (H^2 P^1_{k/k} H^{2^T} + R^2)^{-1} \qquad (5,14)$$

The difference with the hierarchical fusion with full feedback is in the algorithm of the fusion center. In the hierarchical fusion with full feedback the number of sensors is not fixed, but in the hierarchical fusion with reduced communication requirements the number of sensors is fixed (N=2), so an extra sensor will also demand a new fusion algorithm.
The is no need to connect the input u_k to the fusion center because the predicted global state estimate can also be calculated in local filter 1 with the global state estimate that is fed back.

Chapter 6. Distributed model architecture

To reduce the computational load of the previous discussed methods, we looked for a way of fusing reduced order models. The local filters and the fusion centre in the previous stated methods are all of order n. Reduced computational load and reduced bandwidth requirements from sensor nodes to the fusion centre can be reached by reducing the order of the local filters. Subsequently the local states are fused with each other with help of the known models to a global state estimate. This problem is referred to in the literature as the distributed model architecture.

In [6, page 68] a schematically representation for the distributed model fusion is introduced. The architecture looks the same as the architecture of the hierarchical fusion method in chapter 4, but the difference is that the order of the local filters are of order n_i and the fusion centre is of order n. In [10] the problem of data fusion in a distributed architecture is considered.

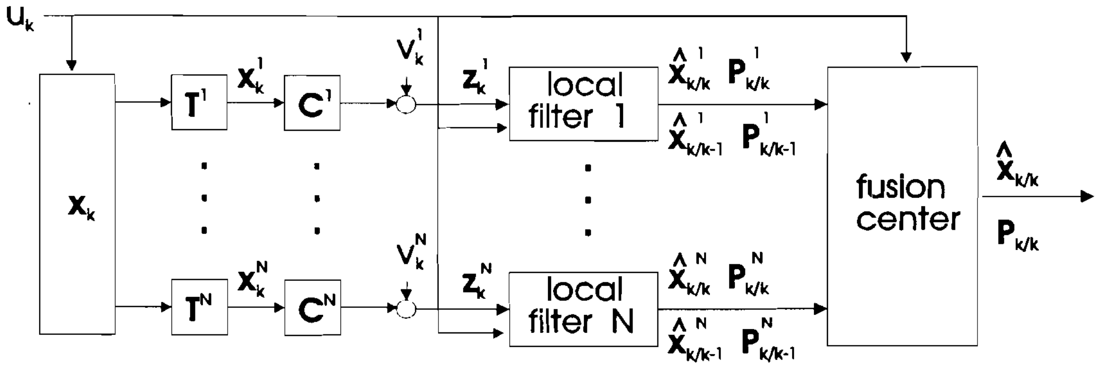

Figure 9, distributed hierarchical fusion architecture

The local state vector $\mathbf{x}_i$ is related to the global state vector $\mathbf{x}$ by the nodal transformation matrices $\mathbf{T}^i$.

$$\begin{aligned} x_k^i &= T_k^i x_k \\ H_k^i &= C_k^i T_k^i \end{aligned} \tag{6,1}$$

The nodal observation equation is based on the global state vector and based on the local state vector.

$$\begin{aligned} z_k^i &= H_k^i x_k + v_k^i \\ z_k^i &= C_k^i T_k^i x_k + v_k^i \\ z_k^i &= C_k^i x_k^i + v_k^i \end{aligned} \tag{6,2}$$

The nodal state transition equation with the nodal state transition matrix and the noise transition matrix is given by the following equation, where the nodal state transition matrix can be obtained from the global state transition matrix using the nodal transformation matrix.

$$\begin{aligned} x^i_{k+1} &= \Phi^i_k x^i_k + \Gamma^i_k w^i_k + \Psi^i_k u^i_k \\ \Phi^i_k &= T^i_{k+1}\,\Phi_k (T^i_k)^- \\ \Gamma^i_k &= T^i_{k+1}\Gamma_k (T^i_k)^- \\ \Psi^i_k &= T^i_{k+1}\,\Psi_k (T^i_k)^- \end{aligned} \tag{6,3}$$

The inverse of the nodal transformation matrix is taken as the Moore-Penrose generalized inverse.

$$(T^i_k)^- = T^{i\,T}_k\,[T^i_k T^{i\,T}_k]^{-1}$$

Every reduced order Kalman filter has two stages, which are prediction:

$$\hat{x}^i_{k/k-1} = \Phi^i_{k-1}\hat{x}^i_{k-1/k-1} + \Psi^i_{k-1}u_k \tag{6,4}$$

$$\hat{P}^i_{k/k-1} = \Phi^i_{k-1}\hat{P}^i_{k-1/k-1}\,\Phi^{i\,T}_{k-1} + \Gamma^i_{k-1}Q^i_{k-1}\Gamma^{i\,T}_{k-1} \tag{6,5}$$

and update:

$$\hat{x}^i_{k/k} = \hat{x}^i_{k/k-1} + P^i_{k/k}C^{i\,T}(R^i)^{-1}\ (z^i_k - C^i\hat{x}^i_{k/k-1}) \tag{6,6}$$

$$(P^i_{k/k})^{-1} = (P^i_{k/k-1})^{-1} + C^{i\,T}(R^i)^{-1}C^i \tag{6,7}$$

The fusion equations consist of a global state estimate equation and a global covariance equation:

$$P^{-1}_{k/k}\hat{x}_{k/k} = P^{-1}_{k/k-1}\,\hat{x}_{k/k-1} + \sum_{i=1}^{N}(T^{i\,T}((P^i_{k/k})^{-1}\hat{x}^i_{k/k} - (P^i_{k/k-1})^{-1}\hat{x}^i_{k/k-1})) \tag{6,8}$$

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + \sum_{i=1}^{N}(T^{i\,T}((P^i_{k/k})^{-1} - (P^i_{k/k-1})^{-1})T^i) \tag{6,9}$$

The predicted global estimate and the predicted global covariance can be computed from the extrapolation equations (3,6) and (3,7).
Similar to the hierarchical fusion algorithm a more convenient equation for the global state estimate is derived

$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + P^{-1}_{k/k}\sum_{i=1}^{N}(T^{i\,T}(P^i_{k/k})^{-1}(\hat{x}^i_{k/k} - T^i\hat{x}_{k/k-1}) + T^{i\,T}(P^i_{k/k-1})^{-1}(T^i\hat{x}_{k/k-1} - \hat{x}^i_{k/k-1})) \tag{6,10}$$

It can be seen that if the transformation matrix is unity, equation (6,10) is similar to the hierarchical fusion equation (4,3).

6.1.Derivation of the distributed fusion algorithm

The global and nodal noise matrices are related via:

$$H^T R^{-1} H = \sum_{i=1}^{N} \left(T^{i\,T} C^{i\,T} (R^i)^{-1} C^i T^i \right)$$

With the help of this relation a global state covariance equation (6,9) can be derived:

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + H^T R^{-1} H$$

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + \sum_{i=1}^{N} \left(T^{i\,T} C^{i\,T} (R^i)^{-1} C^i T^i \right)$$

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + \sum_{i=1}^{N} \left(T^{i\,T} \left((P^i_{k/k})^{-1} - (P^i_{k/k-1})^{-1} \right) T^i \right)$$

The global observations and the local observations are related via:

$$H^T R^{-1} z_k = \sum_{i=1}^{N} \left(T^{i\,T} C^{i\,T} (R^i)^{-1} z^i_k \right)$$

With help of this relation a global state estimate equation (6,8) can be derived:

$$\hat{x}^i_{k/k} = \hat{x}^i_{k/k-1} + P^i_{k/k} C^{i\,T} (R^i)^{-1} (z^i_k - C^i \hat{x}^i_{k/k-1})$$

$$(P^i_{k/k})^{-1} (\hat{x}^i_{k/k} - \hat{x}^i_{k/k-1}) + C^{i\,T} (R^i)^{-1} C^i \hat{x}^i_{k/k-1} = C^{i\,T} (R^i)^{-1} z^i_k$$

$$(P^i_{k/k})^{-1} \hat{x}^i_{k/k} + (C^{i\,T} (R^i)^{-1} C^i - (P^i_{k/k})^{-1}) \hat{x}^i_{k/k-1} = C^{i\,T} (R^i)^{-1} z^i_k$$

$$(P^i_{k/k})^{-1} \hat{x}^i_{k/k} - (P^i_{k/k-1})^{-1} \hat{x}^i_{k/k-1} = C^{i\,T} (R^i)^{-1} z^i_k$$

$$H^T R^{-1} z_k = \sum_{i=1}^{N} T^{i\,T} \left((P^i_{k/k})^{-1} \hat{x}^i_{k/k} - (P^i_{k/k-1})^{-1} \hat{x}^i_{k/k-1} \right)$$

With the rewritten equation (3,4) put in the equation above, we get an equation for the global state estimate.

$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + P_{k/k} H^T R^{-1} (z_k - H \hat{x}_{k/k-1})$$

$$P^{-1}_{k/k} \hat{x}_{k/k} = P^{-1}_{k/k} \hat{x}_{k/k-1} + H^T R^{-1} (z_k - H \hat{x}_{k/k-1})$$

$$= (P^{-1}_{k/k-1} + H^T R^{-1} H) \hat{x}_{k/k-1} + H^T R^{-1} z_k - H^T R^{-1} H \hat{x}_{k/k-1}$$

$$= P^{-1}_{k/k-1} \hat{x}_{k/k-1} + H^T R^{-1} z_k$$

$$= P^{-1}_{k/k-1} \hat{x}_{k/k-1} + \sum_{i=1}^{N} \left(T^{i\,T} \left((P^i_{k/k})^{-1} \hat{x}^i_{k/k} - (P^i_{k/k-1})^{-1} \hat{x}^i_{k/k-1} \right) \right)$$

Now with help of equations (6,8) and (6,9) a more convenient equation for the global state estimate is derived.

$$P^{-1}_{k/k} = P^{-1}_{k/k-1} + \sum_{i=1}^{N} \left(T^{i^T} \left((P^{i}_{k/k})^{-1} - (P^{i}_{k/k-1})^{-1} \right) T^{i} \right)$$

$$P^{-1}_{k/k}\hat{x}_{k/k-1} = P^{-1}_{k/k-1}\hat{x}_{k/k-1} + \sum_{i=1}^{N} \left(T^{i^T} (P^{i}_{k/k})^{-1} T^{i}\hat{x}_{k/k-1} - T^{i^T} (P^{i}_{k/k-1})^{-1} T^{i}\hat{x}_{k/k-1} \right)$$

$$P^{-1}_{k/k-1}\hat{x}_{k/k-1} = P^{-1}_{k/k}\hat{x}_{k/k-1} - \sum_{i=1}^{N} \left(T^{i^T} (P^{i}_{k/k})^{-1} T^{i}\hat{x}_{k/k-1} - T^{i^T} (P^{i}_{k/k-1})^{-1} T^{i}\hat{x}_{k/k-1} \right)$$

$$P^{-1}_{k/k}\hat{x}_{k/k} = P^{-1}_{k/k-1}\hat{x}_{k/k-1} + \sum_{i=1}^{N} \left(T^{i^T} (P^{i}_{k/k})^{-1} \hat{x}^{i}_{k/k} - T^{i^T} (P^{i}_{k/k-1})^{-1} \hat{x}^{i}_{k/k-1} \right)$$

$$P^{-1}_{k/k}\hat{x}_{k/k} = P^{-1}_{k/k}\hat{x}_{k/k-1} - \sum_{i=1}^{N} \left(T^{i^T} (P^{i}_{k/k})^{-1} T^{i}\hat{x}_{k/k-1} - T^{i^T} (P^{i}_{k/k-1})^{-1} T^{i}\hat{x}_{k/k-1} \right) + \\ + \sum_{i=1}^{N} \left(T^{i^T} (P^{i}_{k/k})^{-1} \hat{x}^{i}_{k/k} - T^{i^T} (P^{i}_{k/k-1})^{-1} \hat{x}^{i}_{k/k-1} \right)$$

$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + \\ + P^{-1}_{k/k} \sum_{i=1}^{N} \left(T^{i^T} (P^{i}_{k/k})^{-1} (\hat{x}^{i}_{k/k} - T^{i}\hat{x}_{k/k-1}) + T^{i^T} (P^{i}_{k/k-1})^{-1} (T^{i}\hat{x}_{k/k-1} - \hat{x}^{i}_{k/k-1}) \right)$$

So in this section the fusion equations (6,8), (6,9) and (6,10), for the distributed model architecture that is represented in chapter 6, are derived.

Chapter 7. H_∞ filtering

The aim of this section is define an estimator, dual to the Kalman filters in the previous chapters, that takes into account the frequency dependent properties of a sensor. Such kind of estimator is found in the H_∞ filter. So we try to find a way of designing a H_∞ filter that takes into account the frequency dependent properties of a sensor. We are looking for a way to define a H_∞ filter in similar way the central filter (chapter 3) is defined for fusing measurement data.

We looked in the literature if H_∞ filtering was used in combination with sensor fusion, but no efforts in that direction were found.

Just like the Kalman filter, The H_∞ filter is a causal, linear mapping taking the control input u and the measurement y as its inputs, and producing an estimate $\hat{z}$ of the signal z in such way that the H_∞ norm of the transfer function from the noise w to the estimation error $e=z-\hat{z}$ is minimal. We want to design a filter mapping $(u,y)\rightarrow\hat{z}$ such that the for overall configuration with transfer function $E:(w,v)\rightarrow e$ the H_∞ norm is less than or equal to some pre-specified value γ^2. In this way the H_∞ filter is defined, we need access to the input u.

We consider the state space equations:

$$\begin{aligned} \dot{x} &= Ax + B_1 w + B_2 u \\ z &= C_1 x + D_{21} u \\ y &= C_2 x + v \end{aligned} \qquad (7,1)$$

The H_∞ norm is defined by:

$$\|E\|_\infty^2 = \sup_{w,v\in\mathcal{L}_2} \frac{\|e\|_2^2}{\|w\|_2^2 + \|v\|_2^2} \qquad (7,2)$$

The configuration of the H_∞ is:

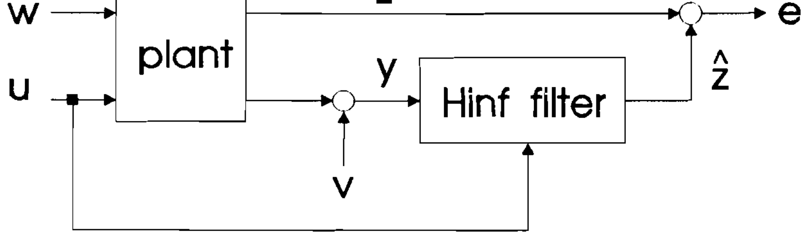

Figure 10, The H_∞ filter configuration

The solution to this problem is given by the following theorem:
Theorem of the H_∞ filter:
There exists a filter which achieves that the mapping E:(w,v)→e in the configuration of Figure 10 satisfies

$$\|E\|_\infty < \gamma$$

if the following Riccati equation has a stabilizing solution $Y=Y^T \geq 0$.

$$0 = AY + YA^T - Y[C_2^T C_2 - \gamma^{-2} C_1^T C_1]Y + B_1 B_1^T \qquad (7,3)$$

In that case one such filter is given by the equation

$$\begin{aligned} \dot{\xi} &= (A + \gamma^{-2} B_1 B_1^T X)\xi + B_2 u + L(C_2 \xi - y) \\ \hat{z} &= C_1 \xi + D_{21} u \\ L &= Y C_2^t \end{aligned} \qquad (7,4)$$

For the H_∞ we assume that the following assumptions hold:
A-1 $D_{11} = 0$ and $D_{22} = 0$.
A-2 The triple (A, B_2, C_2) is stabilizable and detectable.
A-3 The triple (A, B_1, C_1) is stabilizable and detectable.
A-4 $D^T_{12} (\, C_1 \;\; D_{12} \,) = (\, 0 \;\; I \,)$.
A-5 $D^T_{21} (\, B^T_1 \;\; D^T_{21} \,) = (\, 0 \;\; I \,)$.

Assumption A-1 states that there is no direct feedthrough in the transfers w → z and u → z. The second assumption A-2 states that internally stabilizing controllers exist. Assumption A-3 is a technical assumption made on the transfer function. Assumptions A-4 and A-5 are just scaling assumptions that can be easily removed. Further explanation on the assumptions can be found [11, page 64].

The resulting filter is a filter like the Kalman filter but the difference is that there is no real covariance estimated, but the resulting matrix L (the H_∞ filter gain) depends on the value of γ and γ depends on Y.

The computation of H_∞ filters can be done with the various routines included in the Matlab Robust Control Toolbox. For the H_∞ filter design is of importance the Matlab routine hinf and the time discrete variant dhinf.

For the time discrete domain we consider the state space equations:

$$\begin{aligned} x_{k+1} &= A x_k + B_1 w_k + B_2 u_k \\ z_k &= C_1 x_k + D_{12} u_k \\ y_k &= C_2 x_k + v_k \end{aligned} \qquad (7,5)$$

We define $\mathbf{y}_k$, $\mathbf{C}_2$ and $\mathbf{v}_k$ equal to the difinition in chapter 3 to allow several sensor measurements into the system.

$$\begin{aligned} y_k &\triangleq [(y_k^1)^T , \ldots\ldots , (y_k^N)^T]^T \\ C_2 &\triangleq [(C_2^1)^T , \ldots\ldots , (C_2^N)^T]^T \\ v_k &\triangleq [(v_k^1)^T , \ldots\ldots , (v_k^N)^T]^T \end{aligned} \tag{7,6}$$

This is equal to:

$$y_k \triangleq \begin{bmatrix} y_k^1 \\ : \\ y_k^N \end{bmatrix} \quad C_2 \triangleq \begin{bmatrix} C_2^1 \\ : \\ C_2^N \end{bmatrix} \quad v_k \triangleq \begin{bmatrix} v_k^1 \\ : \\ v_k^N \end{bmatrix} \tag{7,7}$$

The discrete time H_∞ filter is now configured by:

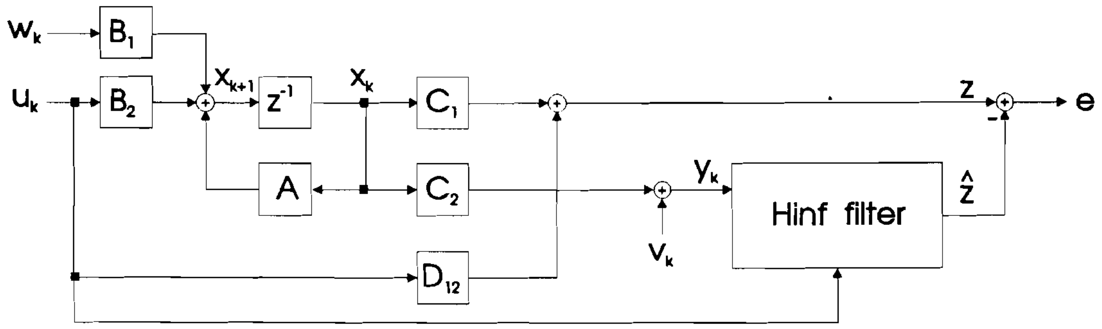

Figure 11, The discrete time H_∞ filter configuration

With this definitions it must can construct a filter with a structure that looks like the central fusion structure. The construction of the H_∞ filter is not carried out yet because we want first look to the simulation results of the previous found fusion algorithms.

Chapter 8. Simulations

In this chapter we carry out simulations with the various fusion algorithms on a simple model with inputs vectors and noise vectors generated by matlab. The reason for using data generated by matlab is that it is easier to compare the fusion algorithms. We want to investigate the effect of various sensor measurements and sensor noises on the fusion algorithms. We are also interested in the effect of bias in a sensor measurement and the possibility of detecting a defect or inaccurate sensor.

8.1. Simulation model

The model used is a vehicle moving in one direction, namely position x. The input vectors of the model are the control input u and the process noise w.

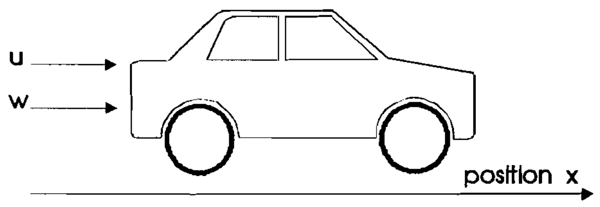

Figure 12, vehicle

The state vector is assumed to consist of position and velocity.

$$\boldsymbol{x} \triangleq \begin{bmatrix} x \\ v \end{bmatrix} \triangleq \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{8,1}$$

The control input u and the process noise w are entered in the system as an acceleration a. where the α is the damping of the system, in our simulations equal to zero.

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ 0 & -\alpha \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{8,2}$$

Measuring the position x is done by:

$$z^x = [1 \quad 0]\, \boldsymbol{x} + v^x \tag{8,3}$$

Measuring the speed v is done by:

$$z^s = [0 \quad 1]\, \boldsymbol{x} + v^s \tag{8,4}$$

Because all the fusion algorithms are derived for the discrete time the model is converted to a discrete time model. This conversion is performed with the matlab conversion program C2DM, which converts the continuous time state-space system to a discrete time state-space system. The conversion method used is a zero order hold. The discrete time state space system is now defined.by (in this case the damping α is equal to zero):

$$x_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} w_k + \begin{bmatrix} \frac{1}{2}T^2 \\ T \end{bmatrix} u_k \qquad (8,5)$$

Where T is the sampling period.
The measurement equations for measuring the position and the speed are the same as in the time continuous case with as only difference the time indices k.

position measurement:

$$z_k^x = [1 \quad 0]\, x_k + v_k^x \qquad (8,6)$$

speed measurement:

$$z_k^s = [0 \quad 1]\, x_k + v_k^s \qquad (8,7)$$

First we make a matlab file (Appendix B: makedat.m) that produces measurement data of the position and the speed and also some sensor noise and with process error. Assumed is that the initial state at t=0 is known, the position and speed at t=0 are equal to zero.

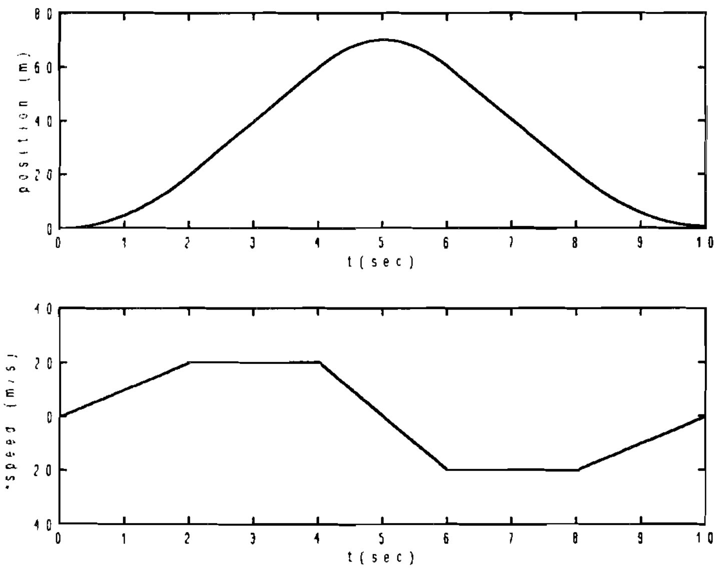

Figure 13, Position and speed of vehicle

From t=0 sec to the input u is a constant acceleration of 10 m/s, then the acceleration is zero for 2 seconds, then the acceleration is -20 m/s for 2 seconds next the acceleration is zero for 2 seconds and the last 2 seconds the acceleration is 10 m/s. The sampling time T=0.2 sec is equal to 10 sec divided by the amount of samples (10 sec / 500 = 0.02 sec).

The syntax for makedat.m is makedat(samples,Q,R1,R2,R3,R4), where samples is the amount of samples taken, Q is the process noise variance and R1 through R4 the variances of four sensor noises which can be added on a position or speed measurement. The m-file makedat.m returns a vector that consists of a position (measure(:,1)) and a speed measurement (measure(2,:) without noise and 4 independent sensor noises with variances given at the input of the m-file. The shape of the return vector is [measure, n1, n2, n3, n4, A, B, C, Q, u]. The m-file resets the seed of the random generation every time it is used, because of this the random noise generators (RANDN) will give the same result and so will not influence the result of the different fusion algorithms.

We made Figure 13 with 500 samples, a process noise variance of 0.1. In the figure the position and the speed are represented without the sensor noise added. Because of this setup we make position and speed measurements by adding sensor noise (n1 to n2) to the observation.

The syntax in the matlab command window is:

```
»[measure,n1,n2,n3,n4,A,B,C,Q,u] = makedat(500,0.1,1,1,1,1);
```

The mean values and variances of the returned noises are:

```
» mean(n1)= -0.0779       var(n1) = 1.1028
» mean(n2)= -0.0174       var(n2) = 1.0272
» mean(n3)=  0.0058       var(n3) = 1.0020
» mean(n4)= -0.0112       var(n4) = 1.1010
```

As seen the mean value of the returned noises are not completely zero and the variances differ a little from each other and the value at the input. This is caused by the finite length of the noise vector, namely 500 samples.

8.2. Two position sensors

In this section we compare the result of the fusion algorithms when we have two position measurements. The measurement equations of the two sensors are given by:

$$\begin{aligned} z_k^1 &= [1 \;\; 0]\, x_k + v_k^1 \\ z_k^2 &= [1 \;\; 0]\, x_k + v_k^2 \end{aligned} \qquad (8,8)$$

With the measurement data generated in section 8.1 we can define the two measurement vectors $\mathbf{z}^1$ and $\mathbf{z}^2$ by adding two uncorrelated noises (e.g. n1 and n2) on the position measurement vector *measure(:,1)*. With the resulting measurement vectors $\mathbf{z}^1$ = (position + n1) and $\mathbf{z}^2$ = (position + n2) we investigate the result of the different fusion algorithms.

First we looked what happens in the 2 position measurements are added and divided by 2. The error n of the fused signal can be calculate with n = ($\mathbf{z}^1 + \mathbf{z}^2$)/2 - measure(:,1). Where *measure(:,1)* represents the original position vector, not corrupted with noise.

The mean value of error n (mean(n)) can be calculated with

$$mean(n) = \frac{\sum_{i=1}^{N} (mean(n_i))}{N}$$

The variance of n (var(n)) can be calculated with

$$var(n) = \frac{\frac{1}{N}\sum_{i=1}^{N} (var(n_i))}{N}$$

In Figure 14 the mean values and the variances of the noises nl, n2 and error n are represented.

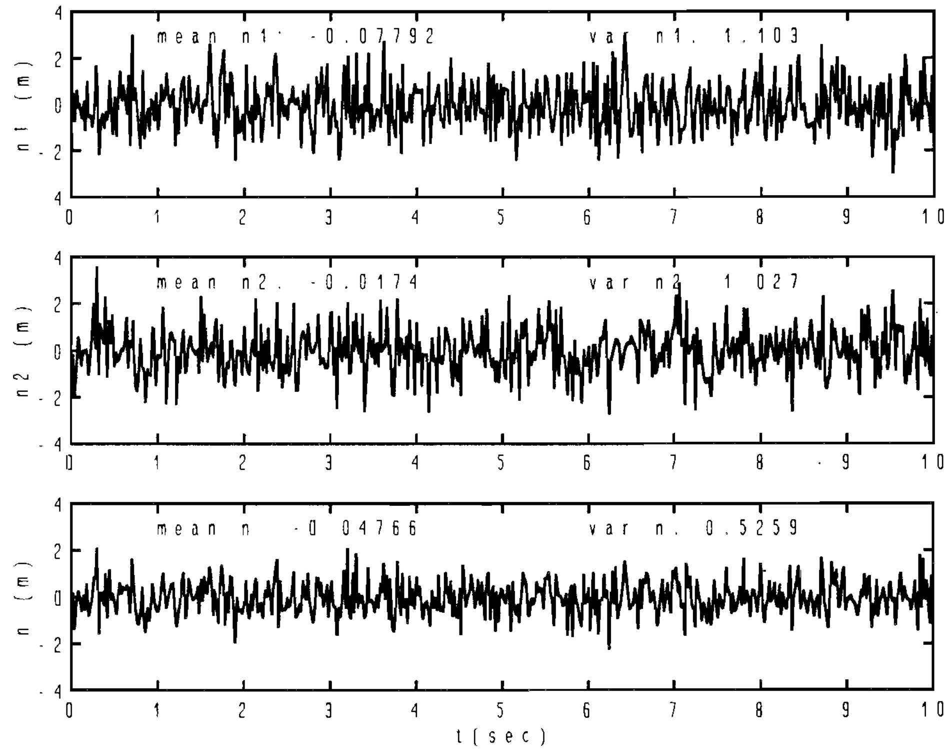

Figure 14, Sensor noises nl and n2 with addition noise

We also looked with matlab to the behaviour of the variance of error n as a function of the proportion of the variance of n1 and the variance of n2. The variance of n1 is set to 1 and the variance of n2 is calculated with a proportion V=var(n2)/var(n1) vary from 1 to 10.

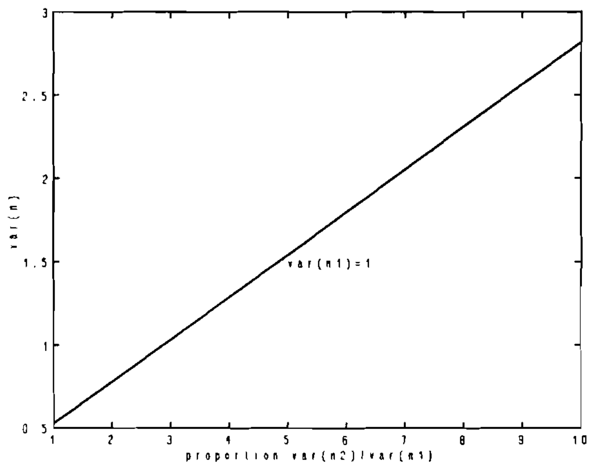

Figure 15, variance n versus proportion V

In Figure 15 is expressed that if the variance of n2 more than 3 times the variance of n1 the resulting error variance becomes bigger than the noise variance of n1, so if one only looks at the variance there is no profit in that case.

8.2.1. Central fusion with 2 position sensors

In this section we investigate the result when the two position measurements z^1 and z^2 are fused with the central fusion method. The equations of the central fusion algorithm are placed in the matlab file central.m and this file is used in the matlab file centr82.m to produce the simulation results that can be compared with the results of the measurement addition in the previous section and the fusion algorithm in the next section. Both the matlab files central.m and centr82.m are presented in appendix C. In Figure 16 the position error after fusing z^1 and z^2 is displayed, the mean value and the variance are calculated after 2 seconds when the covariance matrix Pglob almost reached his final value.

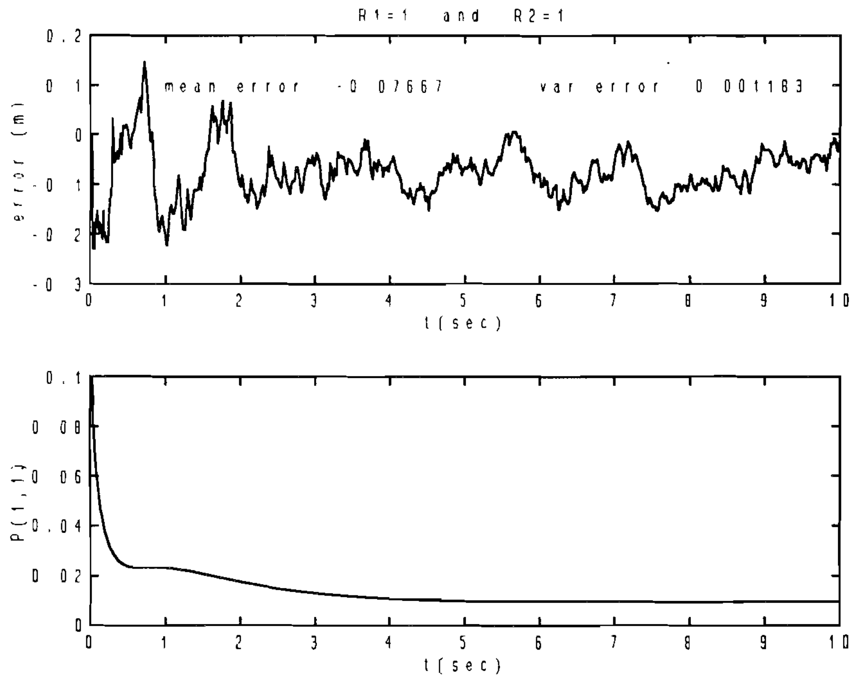

Figure 16, Central fusion with 2 position measurements

You can see that the mean(noise)=-0.07667, this value is larger than the mean value after simple measurement addition (-0.04766). The variance (var(noise)=0.001183) is much smaller than the variance after simple measurement addition (0.5259), this effect is also due to averaging the measurements over more samples. So one can not compare this two results with each other.

Now we investigate what the effect is when the variances of the sensor noises are different. The variance R2 was increased from 1 to 10 and the variance R1 hold to 1. The mean and variance were compared to the mean and variance of a filtered addition measurement, the results are represented in Figure 17. We found that the mean and the variance of the error with the central fusion increased less than the mean and variance of the filtered addition measurement. This is because the central fusion algorithm weights the update of the sensor with the high variance less than the sensor with the low variance in contrast to when both the measurements are just added and divided by 2.

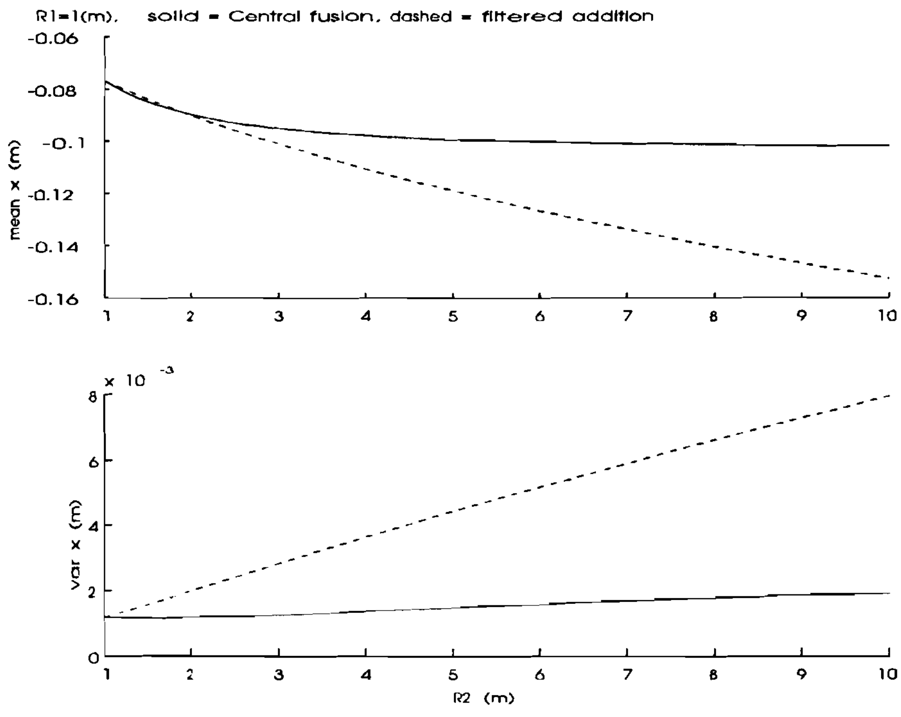

Figure 17, Fusion of sensors with different variances

From Figure 17 we conclude that in the case of two similar sensors with different variances the central fusion will give a smaller mean value of the position error and a smaller variance on the position error than when the signals are added before filtering with a Kalman filter with a known input vector u_k. Maybe a weighted addition, weighted by the sensor variances, before filtering will give the same result as the central fusion algorithm.

We also look what the effect is when a constant value (bias) is added to one of the sensor noises. Therefore a bias from 0.25 m to 10 m is added to one of the measurements and the mean value of the resulting noise after fusion is calculated. The result is displayed in Figure 18 where the bias added to measurement 1 ($z^1 = z^1 +$ bias) is displayed versus the mean value of the error after fusion.

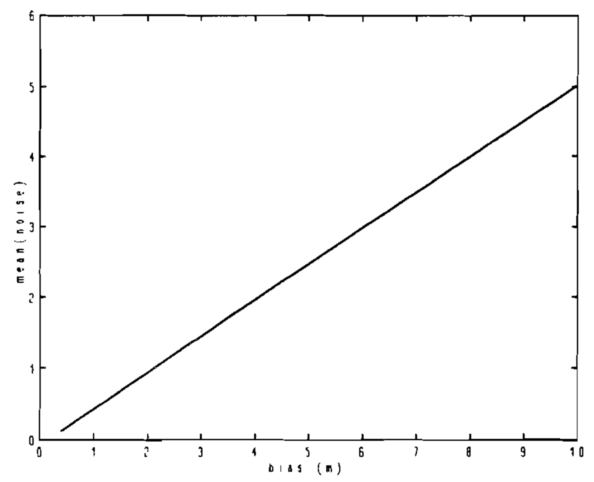

Figure 18, Bias versus mean(noise)

Half the bias added to a measurement returns in the mean value of the noise after the central fusion. This is the same result as if the two measurements are simply added and divided by 2.

The same simulations done with the central fusion algorithm and two position sensors are performed with the other fusion algorithms, namely:

-Hierarchical fusion with two position sensors. (Appendix D)

-Hierarchical fusion with feedback and two position sensors. (Appendix E)

-hierarchical fusion with reduced feedback and two position sensors. (Appendix F)

The results of these simulations are exactly the same as the fusion with the central fusion algorithm. Therefore this results are not represented in this report. At this moment the different fusion algorithms show no difference in the resulting state vectors with different noise variances and different biasses. Therefore we look in the next section how the different fusion algorithm respond to one position measurement and one speed measurement.

8.3. One position sensor and one speed sensor

In this section we compare the result of the fusion algorithms when we have one position measurement and one speed measurement. The measurement equations of the two sensors are given by:

$$z_k^1 = [1 \quad 0]\, x_k + v_k^1$$
$$z_k^2 = [0 \quad 1]\, x_k + v_k^2 \qquad (8,9)$$

With the measurement data generated in section 8.1 we can define the position measurement vector $\mathbf{z}^1$ and speed measurement vector $\mathbf{z}^2$ by adding two uncorrelated noises (e.g. n1 and n2) on the position measurement vector *measure(:,1)* and the speed measurement vector *measure(:,2)*. With the resulting measurement vectors $\mathbf{z}^1$ = (position + n1) and $\mathbf{z}^2$ = (speed + n2) we investigate the result of the different fusion algorithms. As in section 8.2 adding the two position measurements is not possible in this setup, so a comparison with a equivalent method is not possible when fusing a position and a speed measurement.

With some small changes in the files of appendix C through F used in section 8.2.1 we looked at the resulting noise in the fused position and the fused speed vector. Also the mean value and the variance of the noise are calculated. First we investigate if the different fusion algorithms will give the same result if the sensor noises are the same (R1=R2). The position noise and the speed noise and their properties are represented in Figure 19.

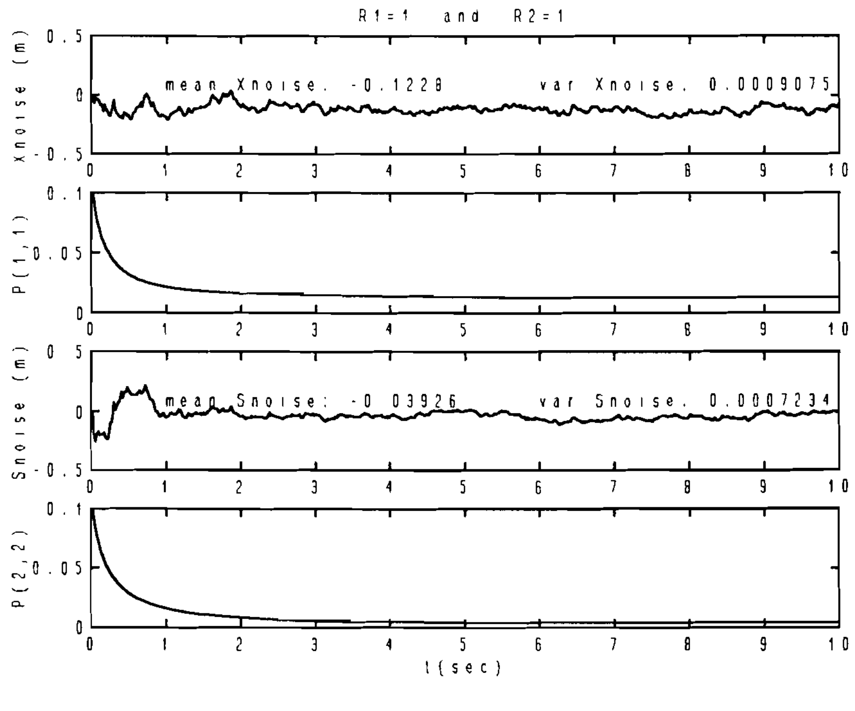

Figure 19, Central fusion with one position and one speed measurement

The simulation of Figure 19 is performed with the central fusion algorithm, the same simulations were performed with the hierarchical fusion, hierarchical fusion with feedback and the hierarchical fusion with reduced feedback fusion algorithms. They all show in the case of R1=R2 exactly the same result.

In Table II we list the result of the simulation in Figure 19 with two other simulations, in one of these simulations the position sensor is very bad (R1=10000) and in the other simulation the speed sensor is very bad (R2=10000). Again the results of all the fusion algorithms are exactly the same. If we look at the difference between the case R1=1/R2=10000 and R1=1/R2=1 we can see that a if we decrease the variance of the speed sensor the mean value of the position error and the mean value of the speed error will increase, and the variance of the position error and the speed error are both decreased. If we look at the difference between the case R1=10000/R2=1 and R1=1/R2=1 we can see that if we decrease the variance of the position sensor the mean value of the position error and the mean value of the speed error will both decrease, and the variance of the position error and the speed error are both decreased. From these simulations we can see that improving one of the sensors will decrease the variances of the fused values. But the mean values not always decreases, this effect can be due to bias in the measurements. This bias is a constant value in the sensor noise caused by the finite length of the noise vector.

Table II, Effect of a fault sensor

	mean position error	mean speed error	variance position error	variance speed error
R1=1, R2=10000	-0.0974	-0.0296	0.003879	0.001888
R1=10000, R2=1	-0.2924	-0.0726	0.009442	0.001044
R1=1, R2=1	-0.1228	-0.0393	0.000908	0.000723

We also looked what the effect of bias is in the case of one position sensor and one speed sensor. In this case we can discern two cases, one when a bias is added on the position measurement ($\mathbf{z}^1 = \mathbf{z}^1 + \text{bias}$) and one when a bias is added on the speed measurement ($\mathbf{z}^2 = \mathbf{z}^2 + \text{bias}$). The simulations were all done with the central fusion algorithm, but we also used the other fusion algorithms and no difference was found. In Figure 20 the mean value of the noise on the position vector and the mean value of the noise on the speed vector are displayed versus a bias from 1 to 10 m/s added on the speed measurement. It can be seen that the effect of a bias on the speed measurement has an effect on the mean value of the position noise and that the effect on the speed noise is decreased. So the position measurement reduces the effect of bias in the speed measurement.

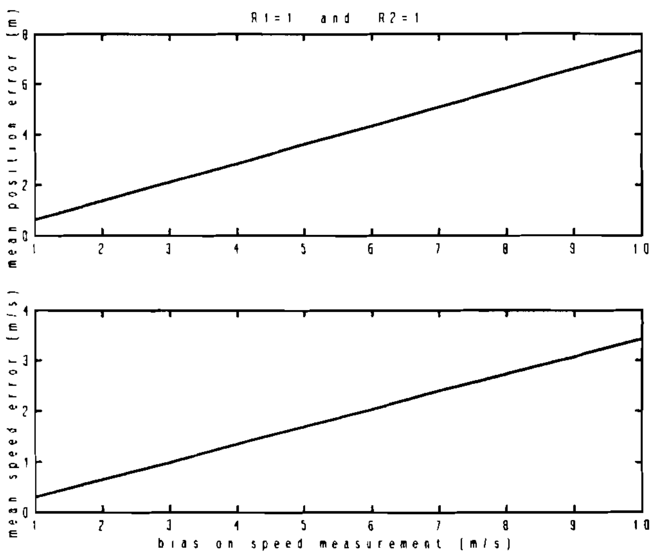

Figure 20, Bias on the speed measurement

In Figure 21 the mean value of the position error and the mean value of the speed error are displayed versus a bias from 1 to 10 m/s added on the position measurement. It can be seen that the bias on the position measurement completely returns in the mean value of the position error. So the mean value of position error is not decreased by the speed measurement. This sounds logic, because the speed is the derivative of the position and so has no information about a constand value in the position measurement.

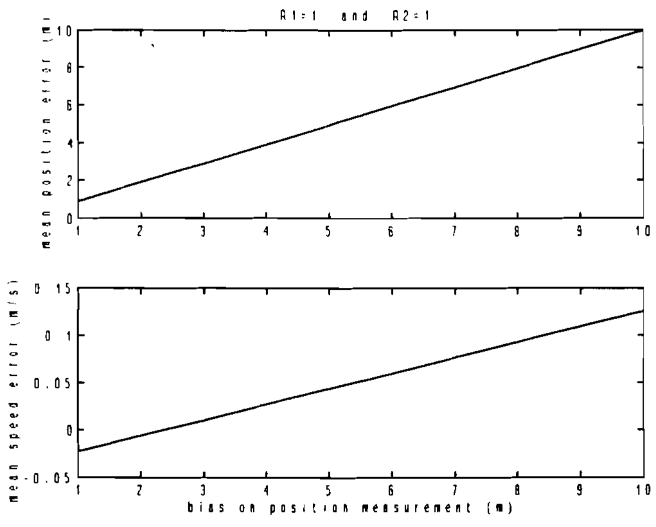

Figure 21, Bias on the position measurement

8.4. multiple position sensors and unknown input u

In section 8.2 and section 8.3 we presented some simulations with the derived fusion algorithms. In this cases the system input vector u_k was assumed to be known. In the projects at our section this system input vector u_k is not known. That is the reason why we did more simulations with multiple position sensors and unknown input vector u_k. We left the input vector u_k out of the fusion algoritme through making the input vector u_k equal to zero after generating the measurement data with *makedat.m*, this is done by adding *u=zeros(size(u));* in the matlab file.

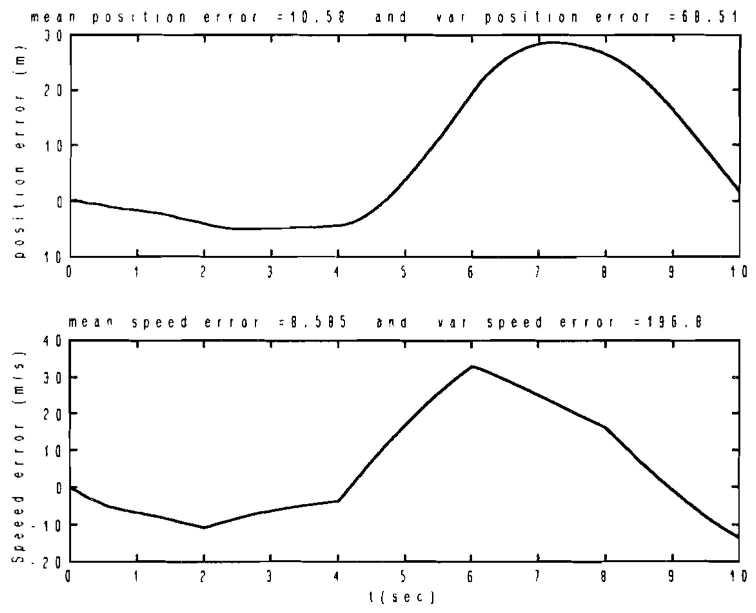

Figure 22, One position sensor, input u_k unknown, Q=0.1

In Figure 22 the position error and the speed error are presented, with one position sensor, when we leave out the input vector u_k in the central fusion algorithm. In the simulation we already increased the amount of samples to 2000 (sampling time T= 10sec/2000 = 5 mSec), but the result is still useless. Further increasing of the number of samples will request too much time for the simulations. In the central fusion algorithm the process noise variance was unchanged (Q=0.1). Because of this the fusion algorithm takes the prediction too much in account. We now investigate how to increase the process noise variance Q, such that the variance of the position error is minimal. The process noise factor Q has to be increased to take into account the unkown input vector $u_k$in the central fusion algorithm.

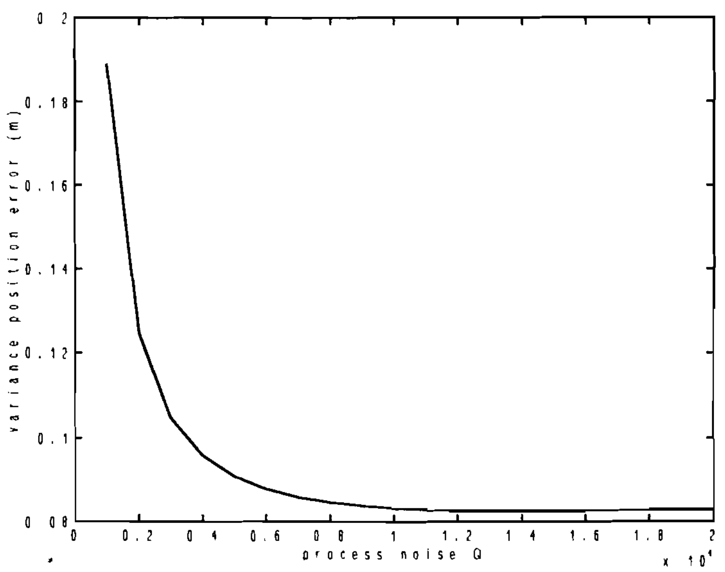

Figure 23, Variance position errror versus process noise Q

From Figure 23 we can determine the Q=14000 that will give a minimum variance on the position error. This value doesn't have to bee exact 14000, and also depends on the used input vector u. Now we do the same simulation as in Figure 22 with the same number of samples (samples = 2000), one position sensor but the process noise Q=14000. In the resulting Figure 24 we see that the resulting position error is improved by increasing the process noise Q in the central fusion algorithm

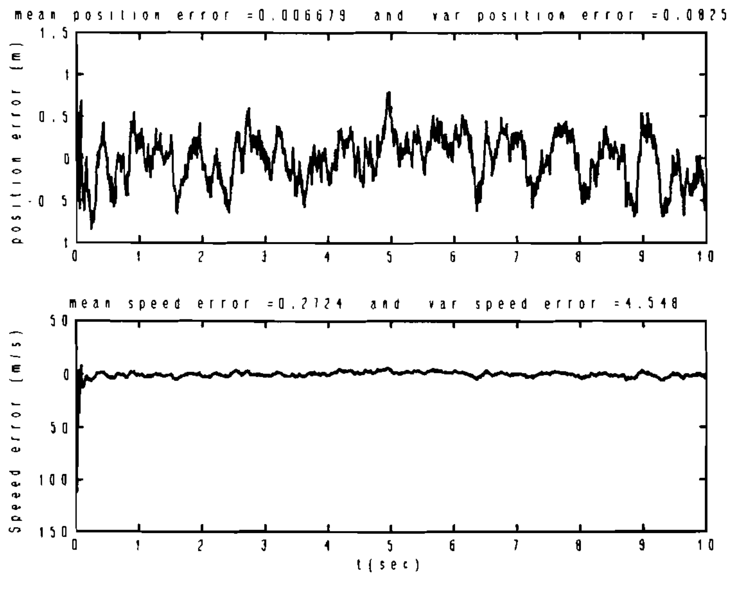

Figure 24, One position sensor, input u_k unknown, Q=14000

Now we have found how we can reach a minimum variance of the position error, we look if we can improve this result by adding more position sensors. Simulations were performed with N=1 to N=8 position sensors, and of each simulation the calculated mean value and variance of the position error and the speed error are presented in table Table III.

Table III, Effect of N position sensors on error

N	mean pos. error	variance pos. error	mean speed error	var.speed error
1	0.00668	0.0825	0.2724	4.548
2	-0.01573	0.0400	0.2161	3.319
3	0.01187	0.0235	0.2143	2.633
4	-0.00016	0.0214	0.2101	2.619
5	0.00002	0.0166	0.2101	2.396
6	0.00175	0.0137	0.2011	2.141
7	0.00136	0.0119	0.1939	2.027
8	0.00237	0.0114	0.1807	2.049

The simulation of eight position sensors is performed again, equivalent to Figure 23, to investigate if the used process noise Q=14000 will give the minimum variance on the position error.

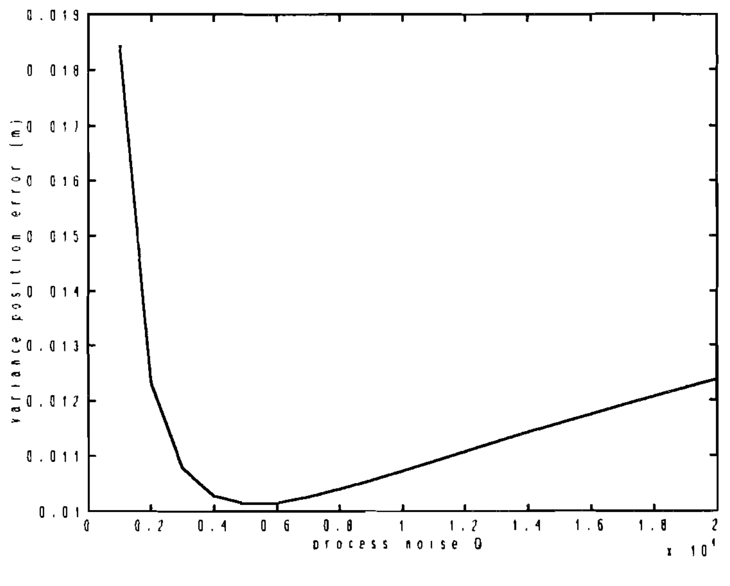

Figure 25, Variance position error versus process noise Q, with N=8

In Figure 25 the variance of the position error is displayed versus the process noise Q. With this figure we can determine that the process noise Q=5000 will give the minimum variance in the position error. If we take Q=5000 the variance in the position error for N=8 position sensors will become 0.0101 meter. The difference of the position error variances 0.0101 and the position error variance calculated with Q=14000 in Table III (0.01143) is only 0.00142 meter.

With the data of Table III we make Figure 26, this figure reflects the position error variance versus number of position sensors N. As concluded from Figure 25, changing the process noise Q will give just a small improvement.

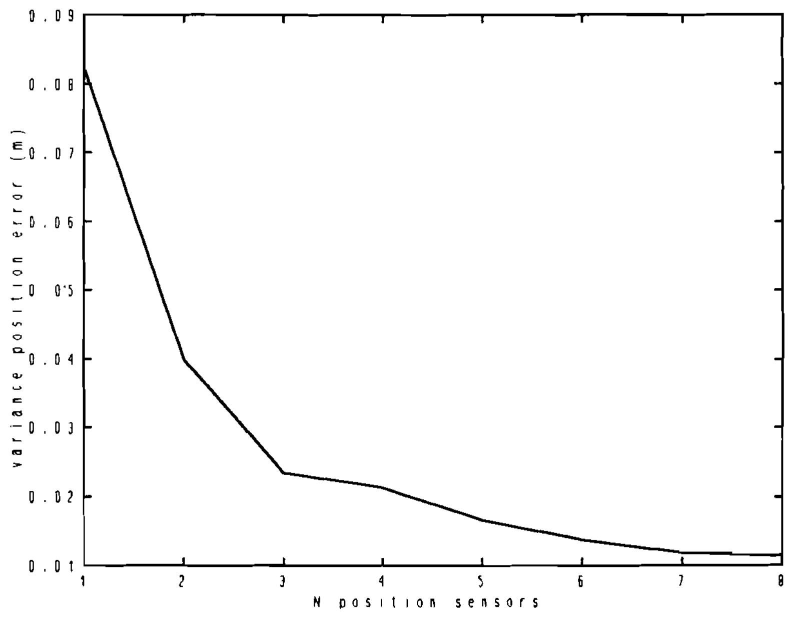

Figure 26, Position error variance versus N position sensors

We looked again to the variance of the postion error as in Figure 26, but with the difference that for sensor 1 we used a speed sensor. One time a speed sensor with variance R1= 1 m/s and one time a speed sensor with variance R1=10 m/s.

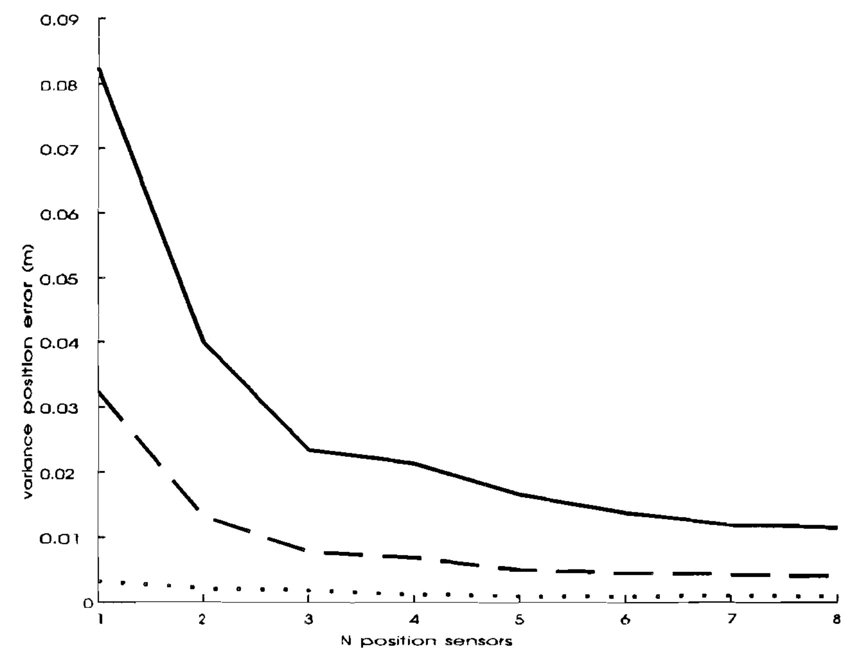

Figure 27, Position error variance, one speed and N-1 postion sensors

In Figure 27 we see that replacing position sensor 1 with R1=1 meter (dashed line) by a speed sensor with R1=10 m/s will give a good improvement in the variance of the position error. Decreasing the variance of this speed sensor to R1=1 m/s (dotted line) will even give a better result.

We now want to know if the used central fusion for 1 to 8 position sensors will give a improvement over the case that the N positions are added before they are filtered. If there is no improvement over the central fusion we could save us a lot of calculation time by first adding the measurements before filtering filtering. The addition is done by:

$$\text{addition measurement} = \frac{\sum_{i=1}^{N} \text{measurement } i}{N}$$

The variance R_{addition} , for filtering the addition, used in the central filter is calculated with:

$$R_{addition} = \frac{\frac{1}{N}\sum_{i=1}^{N} R_i}{N}$$

Now we added the N position measurements and devide them by N. This new measurement is filtered by a central fusion algorithm for one sensor. The results of this are mean values of position errors, position error variances, mean values of speed errors and speed error variances, and they are exactly the same as the ones displayed in Table III acquired by the central fusion algorithm. To show the result is exactly the same we compared the position error of N=8 central fusion with the position error of the filtered addition of 8 position measurements. This comparison is done by subtracting the position error of the N=8 position sensor central fusion result from the N=8 position measurements added before filtering. In Figure 28 we can see that the difference is maximal $5*10^{-14}$, so this is almost zero exept a small error due to computer calculation accuracy.

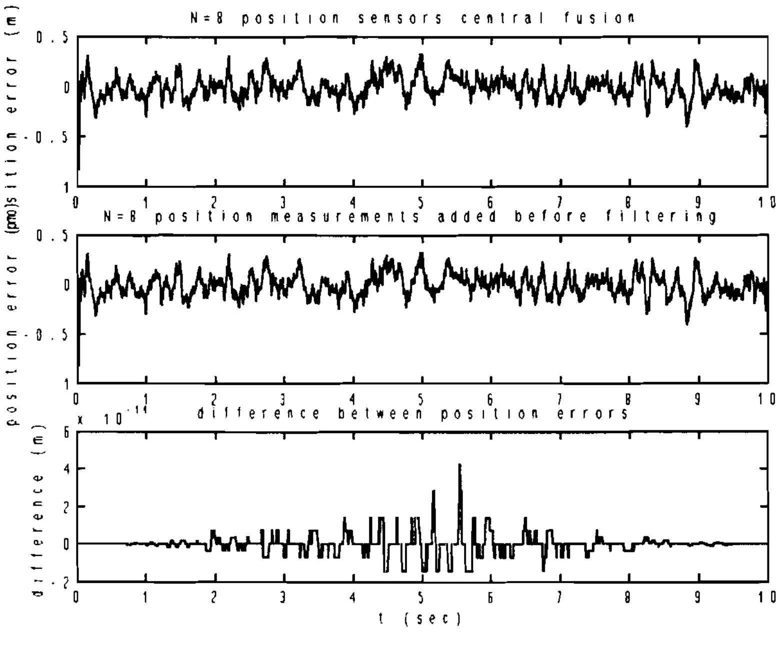

Figure 28, Difference central fusion and filtered addition, N=8

So we can save a lot of calculation time if we just add the N position measurements, with equal variances, before filtering. This is in the special case that all the sensor noise variances are exactly the same, Figure 17 already showed that if the sensor variances are different the central fusion algorithm gives a lower variance on the position error.

In Figure 29, we compare the position error with N=8 position sensors fused with central fusion algorithm and fused hierarchical fusion algorithm. Again there very small difference caused by calculation accuracy. The strange schape of the difference can be explained by the looking at the shape of position x in Figure 13. If the x becomes larger the calculation errors also grow.

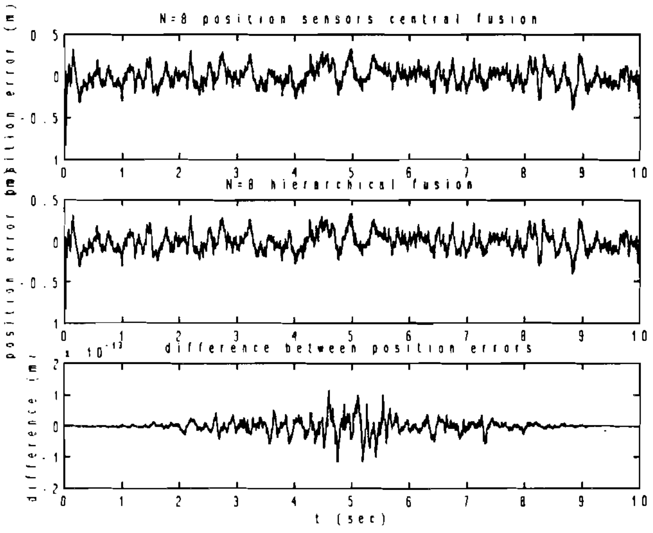

Figure 29, Difference central fusion with hierarchical fusion for N=8

In Figure 32 we investigate the difference between the central fusion and the hierarchical fusion with feedback. Again the result of both the fusion algorithms is the same.

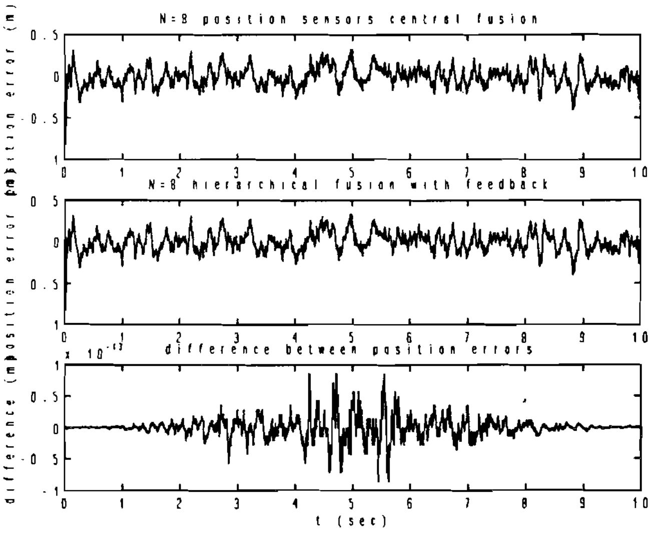

Figure 30, Difference central fusion with hierarchical fusion with feedbak for N=8 position sensors.

8.5. Detecting a fault sensor

In this section we investigate if it is possible to detect, with help of the global measurement and a sensor measurement, a fault sensor. In Figure 31 we show how a error signal with the global state estimate and the sensor measurement is defined in the central fusion scheme.

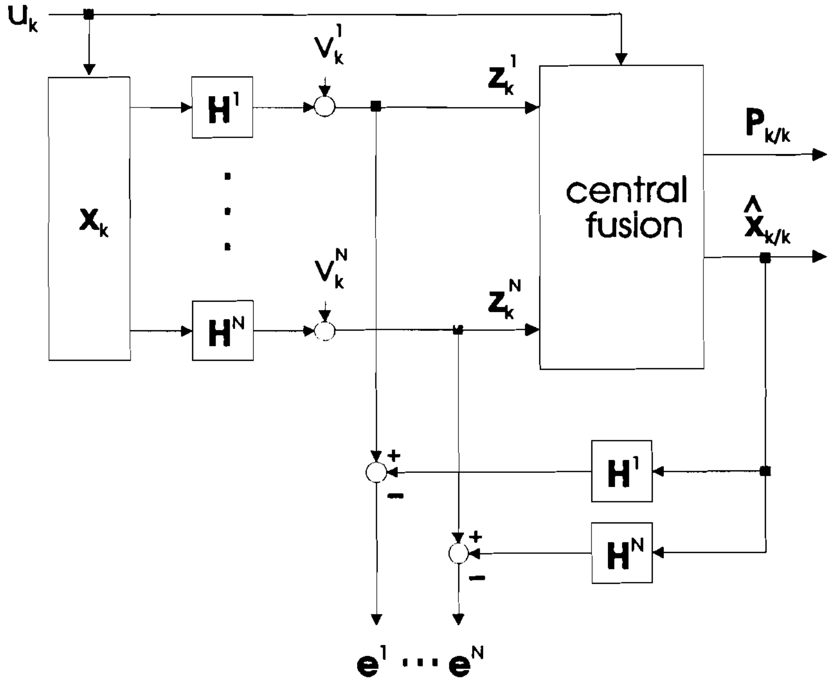

Figure 31, Error signal definition for fault sensor detection

The error signal e^i is defined by:

$$e_k^i = z_k^i - H^i x_{k/k} \qquad (8,10)$$

We now do a simulation with the same one dimensional dynamic model as used in the other sections, with unknown input vector u_k , three position sensors and 2000 samples (T=10 sec / 2000 samples = 5 mSec). We will disrupt one of the position measurements with noise and bias, whereafter we look at the effect on the error signal e^i. In Figure 32 the error signal that is added on the position measurement of sensor 2 is displayed, also the three corresponding error signals according to equation (8,10) are displayed. The error signal is a constand value (bias) of 10 meter added from t=2 to t=4 seconds, and a zero mean noise with variance R=4 meter added from t=6 to t=4 seconds.

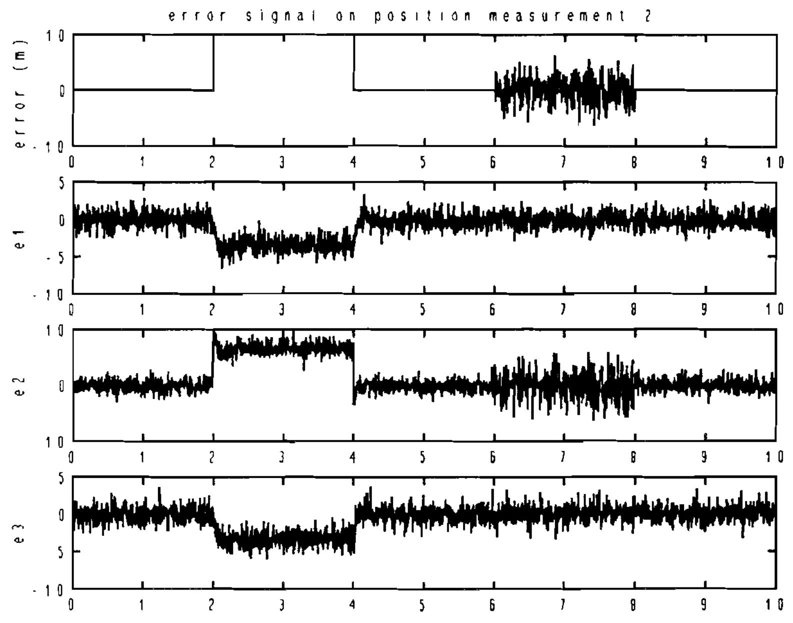

Figure 32, Error signals of three position measurements

It can be seen from the figure that the bias on position measurement 2 has infuence on all three the error signals. But the effect is larger in the error signal of measurement 2 than on the other measurements. The zero mean white noise is also visible in the error signal of position measurement 2. From Figure 32 can be seen that the effect extra bias or extra noise on a sensor measurement can be detected from the defined error signal. To do a correct fault sensor detection, a norm for this has to be defined.

Chapter 9. Conclusions

A simple way in fusing sensor measurements is found in the central fusion architecture. If the sensors process their data locally, and sending their estimates to a central processor, alternative fusing algorithms are needed. A solution for this is found in hierarchical fusion algorithm. The hierarchical fusion algorithm is compared algebraical to the central fusion architecture in the two similar sensor case, which yields the same results. In the litarature was stated that an advantage of the hierarchical fusion algorithms is that it is easy to detect sensor failure by comparing the local estimate with the global estimate. This is not completely correct, because with a defined error signal e, it must also be possible to detect a fault sensor in the central fusion case. A disadvantage is the growing computational load, because the same system model as the central fusion node is used in every local node.
The global state estimate in the hierarchical architecture can also be fed back to the local nodes to be used as priori statistics. This method is also compared algebraical to the central fusion architecture in the two similar sensor case, which yields the same results. The feedback of the globel state estimate and the global covariance yields extra communication requirements, these communication can by reduced by using the algorithm derived in [8, Alouani-AT]. This algorithm is only derived for the 2 sensor case. The hierarchical architectures with and without feedback are both derived from the central architecture. They are also both compared algebraical with the central architecture in the two similar sensor case, from which we conclude that in this case the result of the three architectures are the same.

Because of the growing computational load with hierarchical fusion algorithm we looked for a way to reduce the order of the local filters (nodes). This approach, referred to as the distributed model architecture, has the advantage of reduced computational load at the local nodes and reduced communication bandwidth, compared to the hierarchical fusion architecture.

The central fusion, the hierarchical fusion, the hierarchical fusion with feedback and the hierarchical fusion with reduced feedback are also tested with a one dimensional dynamic model. This tests with data generated with matlab showed that all the results of the fusion algorithms are exactly the same, exept a small difference in the order of 10^{-14} that is a result of calculation accuratie of the computer. The differences between position errors with cental fusion, the hierarchical fusion, hierarchical fusion with feedback and hierarchical fusion with reduced feedback are compared in the various cases, namely:

- Two position sensors, the same sensor noise variance
- Two position sensors, with different sensor noise variance
- One position sensor and one speed sensor, with different sensor noise variances
- Eight position sensors, with different sensor noise variances
- Various position sensors and various speed sensors, with different sensor noise variances (number of sensors N=8).

In all this cases no difference between the global position estimate was detected. Explanation for the fact that al the fusion algorithms give exactly the same result, is that the fusion algorithms are all derived from, and under the same conditions, than the central fusion algorithm.

The simulation with the one dimensional dynamic model are done with and without knowledge of the system input vector u_k. If the system input vector is known the resulting variance on the posion error is smaller than when the system input vector is unknown. Explanation for this is when the system input vector is known, we can make a better prediction of the system state vector and so the gain of the Kalman filter can be smaller. If the system vector u_k is unkown, the influence of his error has to be taken into account into the system noise variance (Q). The system noise variance Q in the fusion algorithm has to be increased so that the Kalman gain won't become too small, and so the fusion algorithm is able to follow fast changes of the system caused by the input vector u_k.

Because all fusion algorithms show us the same result, fusion with the central fusion algorithm is prefered over the other fusion algorithms because the central fusion algorithm uses less calculation time than the other fusion algorithms. We also looked what happens if we add sensor measurements, of the same quantity, before filtering by the central fusion algorithm. Found is that if the variances of the sensor noises are all equal the resulting variance of the position error is the same, compared to central fusion. So in this case we could save us a lot of time by just adding the sensor measurements before filtering. In the case the sensor variances are not the same the variance of the position error with central fusion is smaller, compared to the case when we first add the sensor measurements before filtering.

The simulations showed that the fusion algorithms are usefull when we want to fuse sensor measurement of different quantities, for example in our simulations the quantities position and speed. Also adding sensor measurements of the same quantity will decrease the variances of the state estimate. This effect shows an inverse square effect, so the adding of extra sensors of the same quantity has not a endless positive effect, because it also uses extra calculation time.

Recommendations:

-The fusion algorithms can to be tested with models of larger order. In the case of a larger model one could also pay attention to how a set of sensors should be selected.

-In this report a proposition is done to solve the sensor fusion problen with a H_∞ filter in a equivalent way as the central filter. The H_∞ filter has to be completed, and than the results can be compared with the fusion algorithms described in this report. Expected advantage of the H_∞ filter is that the H_∞ filter could be able to take into account a coloured spectrum of the sensor noise, in contrast to the Kalman filter that assumes white noise with zero mean and known variance.

References

[1] Waltz, E. and J. Llinas
Multisensor data fusion.
Norwood: Artech House, 1990.
The Artech House radar library.
ISBN 0-89006-277-3, Elektro bibl.: LKT 90 MAL

[2] Multisensor fusion and integration for intelligent systems.
Proc. 1994 IEEE int. conf. MFI '94,
Las Vegas, NV, USA, 2-5 oct. 1994.
New York, NY, USA: IEEE, 1994.
ISBN 0-7803-2072-7, Elektro bibl.: DTE 94 MUL

[3] Maybeck, P.S.
Stochastic models, estimation, and control, volume 1
New York: Academic Press, 1979.
Mathematics in science and engineering, vol 141.
ISBN 0-12-480701-1, Elektro bibl.: DTP 79 MAY

[4] Bar-shalom, Y. and T.E. Fortmann
Tracking and data association.
New York: Academic Press, 1988.
ISBN 0-12-079760-7, Bibl. IPO: EF 32

[5] Bar-shalom, Y.
Multitarget-multisensor tracking: Advanced aplications, vol 1.
London: Artech House, 1990.
The Artech House radar library.
ISBN 0-89006-377-X, Elektro bibl.: LTK 90 MUL

[6] Bar-shalom, Y.
Multitarget-multisensor tracking: Applications and advances, vol 2.
London: Artech House, 1992.
The Artech House radar library.
ISBN 0-89006-517-9, Elektro bibl.: MAB 92 MUL

[7] Blackman, S.S.
Multiple target tracking with radar applications. p. 380-393.
Norwood: Artech House, 1986.
The Artech House radar library.

[8] Alouani, A.T. and T.R. Rice, R.E. Helmick
On sensor track fusion
In: Proc. of the 1994 American Control Conf. Baltimore, MD, USA, 29 June-1 July 1994.
New York, NY, USA: IEEE, 1994, Vol. 1, p. 1042-6.
ISBN: 0 7803 1783 1, Elektro bibl.: DPC 82 AME

[9] Hongyan Sun and Wenlong Hu, Pinxing Lin, Shiyi Mao
A study on an algorithm of multisensor data fusion.
In: Proc. of the IEEE 1994 National Aerospace and Electronics Conference NAECON 1994
Dayton, OH, USA, 23-27 May 1994.
New York, NY, USA: IEEE, 1994, Vol. 1, p. 239-45.
ISBN: 0 7803 1893 5, Elektro bibl.: MCM 70 AER

[10] Berg, T.M. and H.F. Durrant-Whyte
Model distribution in decentralized multi-sensor data fusion.
In: Proc. of the 1991 American Control Conference. Boston, MA, USA, 26-28 June 1991.
Evanston, IL, USA: American Autom. Control Council, 1991, vol. 3, p. 2292-3.

[11] Siep Weiland
Robust control
Research report of Eindhoven University of Technologie
Draft version of May 28, 1996

Appendix A: kost.m

Calculation of the proportion V as a function of system order n in section 3.1

```
for N=1:5
 for n=1:10
   A=(2*n^3+2*n^2+n+2*N*n);
   B=(2*N^2*n+2*N*n^2+14/10*n^3);
   C=(3*N*n^2+2*N^2*n+7/10*N^3+n^3);
   V(N,n)=(A+C)/(A+B);
 end
end
hold
plot(V(1,:))
plot(V(2,:))
plot(V(3,:))
plot(V(4,:))
plot(V(5,:))
plot([0:10],ones(11,1),'--')
text(0.2,V(1,1),'N=1')
text(0.2,V(2,1),'N=2')
text(0.2,V(3,1),'N=3')
text(0.2,V(4,1),'N=4')
text(0.2,V(5,1),'N=5')
axis([0 10 0.8 2.2])
XLABEL('order n')
YLABEL('proportion V')
% TITLE('Calculation time proportion method 1 / method 2')
print -dhpgl kost
```

Appendix B: makedat.m

Production of measurement data in section 8.1 of a vehicle moving in one direction

```
% Produces measurement data according to chapter 8.1

function[measure,snoise1,snoise2,snoise3,snoise4,A,B,C,Q,u] = makedat(samples,Q,R1,R2,R3,R4)

% Definitions
time=10;
T=time/samples;

% Reset the seed to its startup value
randn('seed',0)

A=[1 T
  0 1];

% B1 presents the process noise
B1=[0.5*T^2 ; T];

% B2 presents the control input
B2=[0.5*T^2 ; T];

B=[B1 B2];

C=[1 0; 0 1];

D=[0 0 ;0 0];

X=zeros(2,1);

w=sqrt(Q)*randn(samples,1);

% define the input vector u
u=zeros(samples,1);
u(1:samples*0.2)=10*ones(samples*0.2,1);
u(samples*0.4+1:samples*0.6)=(-20)*ones(samples*0.2,1);
u(samples*0.8+1:samples)=10*ones(samples*0.2,1);

U=[w  u];

X0=zeros(2,1);
t=[1:samples];

snoise1=sqrt(R1)*randn(samples,1);
snoise2=sqrt(R2)*randn(samples,1);
snoise3=sqrt(R3)*randn(samples,1);
snoise4=sqrt(R4)*randn(samples,1);

subplot(2,1,1)
[measure,X]=DLSIM(A,B,C,D,U,X0);
plot(t*time/samples,measure(:,1))
xlabel('t(sec)')
ylabel('position (m)')
```

```
subplot(2,1,2)
plot(t*time/samples,measure(:,2))
xlabel('t(sec)')
ylabel('speed (m/s)')
V=axis; V(3)=-40; axis(V);

print -dhpgl g:\verslag\chap81
```

Appendix C: central.m and centr82.m

Matlab function containing the central fusion algorithm as described in this report
central.m:

```
% Central fusion, chapter 3 for N sensors

function[Xglob,Pglob,Xpred,Ppred]=central(A,B,H,Q,u,R,Z,Xinit,Pinit)

RINV=inv(R);

% calculations according to equations 3.6 and 3.7
Xpred=A*Xinit+B(:,2)*u;
Ppred=A*Pinit*A'+B(:,1)*Q*B(:,1)';

% calculation according to equation 3.5
Pglob=inv(inv(Ppred)+H'*RINV*H);

% calculation according to equation 3.4
Xglob=Xpred+Pglob*H'*RINV*(Z-H*Xpred);
```

Matlab file using function central.m in the 2 position sensor case.
centr82.m:

```
samples=500;
Q=0.1;
R1=1;
R2=1;
R3=1;
R4=1;
[measure,n1,n2,n3,n4,A,B,C,Q,u] = makedat(samples,Q,R1,R2,R3,R4);
Z1=measure(:,1)+n1;
Z2=measure(:,1)+n2;

Z=[Z1 Z2]';
H=[1 0; 1 0];
R=diag([R1 R2]);

Xinit=zeros(size(A,1),1);
Pinit=diag([Q Q]);
P(1)=Q;
X(1)=0;

for i=2:samples
  [Xglob,Pglob,Xpred,Ppred] = central(A,B,H,Q,u(i-1),R,Z(:,i),Xinit,Pinit);
  Xinit=Xglob;
  Pinit=Pglob;
  X(i)=Xglob(1);
  P(i)=Pglob(1,1);
end

noise=X'-measure(:,1);
%mean(noise(100:samples))
%var(noise(100:samples))

subplot(2,1,1)
```

```
plot([1:500]*0.02,noise)
Title(['R1=',num2str(R1),' and R2=',num2str(R2)])
xlabel('t(sec)')
ylabel('noise (m)')
text(1,0.1,['mean noise: ',num2str(mean(noise(100:samples)))])
text(6,0.1,['var noise: ',num2str(var(noise(100:samples)))])

subplot(2,1,2)
plot([1:500]*0.02,P)
xlabel('t(sec)')
ylabel('P(1,1)')

%print -dhpgl g:\verslag\chap822
```

Appendix D: hier.m and hier82.m

Matlab function containing the hierarchical fusion algoritm as described in this report

hier.m

```
% Hierarchical fusion, chapter 4 for N sensors

function[Xglob,Pglob]=hier(A,B,Q,u,Xinit,Pinit,Ploc,Plocpred,Xloc,Xlocpred)

n=size(Ploc,2);
N=size(Ploc,1)/size(Ploc,2);

% calculations according to equations 3.6 and 3.7
Xpred=A*Xinit+B(:,2)*u;
Ppred=A*Pinit*A'+B(:,1)*Q*B(:,1)';

% calculation according to equation 4.2
Pglob=inv(Ppred);
for i=1:N
  Pglob=Pglob+(inv(Ploc((i-1)*n+1:i*n,:)) - inv(Plocpred((i-1)*n+1:i*n,:)));
end
Pglob=inv(Pglob);


% calculation according to equation 4.3
Xglob=Xpred;
for i=1:N
  Xglob=Xglob+Pglob*(inv(Ploc((i-1)*n+1:i*n,:))*(Xloc((i-1)*n+1:i*n,:)-Xpred));
  Xglob=Xglob+Pglob*(inv(Plocpred((i-1)*n+1:i*n,:))*(Xpred-Xlocpred((i-1)*n+1:i*n,:)));
end
```

Matlab file using function hier.m in the 2 position sensor case.

hier82.m

```
samples=500;
Q=0.1;
R1=1;
R2=1;
R3=1;
R4=1;
[measure,n1,n2,n3,n4,A,B,C,Q,u] = makedat(samples,Q,R1,R2,R3,R4);
Z1=measure(:,1)+n1;
Z2=measure(:,1)+n2;

X1init=zeros(size(A,1),1);
P1init=diag([Q Q]);
X2init=zeros(size(A,1),1);
P2init=diag([Q Q]);
Xinit=zeros(size(A,1),1);
Pinit=diag([Q Q]);
X(1)=0;

for i=2:samples
  %local filter 1
  [X1,P1,X1pred,P1pred]=central(A,B,[1 0],Q,u(i-1),R1,Z1(i),X1init,P1init);
  X1init=X1;
  P1init=P1;

  %local filter 2
  [X2,P2,X2pred,P2pred]=central(A,B,[1 0],Q,u(i-1),R2,Z2(i),X2init,P2init);
  X2init=X2;
  P2init=P2;

  Xloc=[X1; X2];
  Ploc=[P1; P2];
  Xlocpred=[X1pred; X2pred];
  Plocpred=[P1pred; P2pred];

  %fusion of local filters with hier.m
  [Xglob,Pglob]=hier(A,B,Q,u(i-1),Xinit,Pinit,Ploc,Plocpred,Xloc,Xlocpred);
  Xinit=Xglob;
  Pinit=Pglob;
  X(i)=Xglob(1);
end

noise=X'-measure(:,1);
%mean(noise(100:samples))
%var(noise(100:samples))

subplot(2,1,1)
plot([1:500]*0.02,noise)
Title(['R1=',num2str(R1),' and R2=',num2str(R2)])
xlabel('t(sec)')
ylabel('noise (m)')
text(1,0.1,['mean noise: ',num2str(mean(noise(100:samples)))])
text(6,0.1,['var noise: ',num2str(var(noise(100:samples)))])

subplot(2,1,2)
plot([1:500]*0.02,P)
xlabel('t(sec)')
ylabel('P(1,1)')
```

Appendix E: hierfeed.m and hierfe82.m

Matlab function containing the hierarchical fusion algorithm with feedback as described in this report

hierfeed.m

```
% Hierarchical fusion with feedback, chapter 5 for N sensors

function[Xglob,Pglob]=hierfeed(A,B,Q,u,Xinit,Pinit,Ploc,Plocpred,Xloc,Xlocpred)

n=size(Ploc,2);
N=size(Ploc,1)/size(Ploc,2);

% calculations according to equations 3.6 and 3.7
Xpred=A*Xinit+B(:,2)*u;
Ppred=A*Pinit*A'+B(:,1)*Q*B(:,1)';

% calculation according to equation 5.2
Pglob=(1-N)*inv(Ppred);
for i=1:N
  Pglob=Pglob+inv(Ploc((i-1)*n+1:i*n,:));
end
Pglob=inv(Pglob);


% calculation according to equation 5.3
Xglob=Xpred;
for i=1:N
  Xglob=Xglob+Pglob*inv(Ploc((i-1)*n+1:i*n,:))*(Xloc((i-1)*n+1:i*n,:)-Xpred);
end
```

Matlab file using function hierfeed.m in the 2 position sensor case.

hierfe82.m

```
samples=500;
Q=0.1;
R1=1;
R2=1;
R3=1;
R4=1;
[measure,n1,n2,n3,n4,A,B,C,Q,u] = makedat(samples,Q,R1,R2,R3,R4);
Z1=measure(:,1)+n1;
Z2=measure(:,1)+n2;

X1init=zeros(size(A,1),1);
P1init=diag([Q Q]);
X2init=zeros(size(A,1),1);
P2init=diag([Q Q]);
Xinit=zeros(size(A,1),1);
Pinit=diag([Q Q]);
X(1)=0;

for i=2:samples
  %local filter 1
  [X1,P1,X1pred,P1pred]=central(A,B,[1 0],Q,u(i-1),R1,Z1(i),X1init,P1init);
  %local filter 2
  [X2,P2,X2pred,P2pred]=central(A,B,[1 0],Q,u(i-1),R2,Z2(i),X2init,P2init);

  Xloc=[X1; X2];
  Ploc=[P1; P2];
  Xlocpred=[X1pred; X2pred];
  Plocpred=[P1pred; P2pred];

  %fusion of local filters with hier.m
  [Xglob,Pglob]=hierfeed(A,B,Q,u(i-1),Xinit,Pinit,Ploc,Plocpred,Xloc,Xlocpred);

  Xinit=Xglob;
  Pinit=Pglob;

  %feedback of global state estimate and global covariance to the local
  %estimators, according to equations 5.4 and 5.5
  X1init=Xglob;
  P1init=Pglob;
  X2init=Xglob;
  P2init=Pglob;

  X(i)=Xglob(1);
end

noise=X'-measure(:,1);
%mean(noise(100:samples))
%var(noise(100:samples))

subplot(2,1,1)
plot([1:500]*0.02,noise)
Title(['R1=',num2str(R1),' and R2=',num2str(R2)])
xlabel('t(sec)')
ylabel('noise (m)')
text(1,0.1,['mean noise: ',num2str(mean(noise(100:samples)))])
text(6,0.1,['var noise: ',num2str(var(noise(100:samples)))])

subplot(2,1,2)
plot([1:500]*0.02,P)
xlabel('t(sec)')
ylabel('P(1,1)')
```

Appendix F: hierred.m and hierre82.m

Matlab function containing the hierarchical fusion algorithm with reduced feedback as described in this report.

hierred.m

```
% hierarchical fusion with reduced feedback, chapter 5.3 for 2 sensors

function[Xglob,Pglob]=hierred(A,B,H,Q,u,R2,Xinit,Pinit,Ploc,Plocpred,Xloc,Xlocpred)

n=size(Ploc,2);
N=2;

% calculation equation 5.14
Kfusion=Ploc(1:n,:)*H(2,:)'*inv(H(2,:)*Ploc(1:n,:)*H(2,:)'+R2);

% calculation equation 5.13
Pglob=(eye(n)-Kfusion*H(2,:))*Ploc(1:n,:);

% calculation equation 5.12
Xglob=Pglob*(inv(Ploc(1:n,:))*Xloc(1:n,:)+inv(Ploc(n+1:2*n,:))*Xloc(n+1:2*n,:)-(inv(Ploc(n+1:2*n,:))-H(2,:)'*inv(R2)*H(2
,:))*Xlocpred(n+1:2*n,:));
```

Matlab file using function hierred.m in the 2 position sensor case.

hierre82.m

```
samples=500;
Q=0.1;
R1=1;
R2=1;
R3=1;
R4=1;
[measure,n1,n2,n3,n4,A,B,C,Q,u] = makedat(samples,Q,R1,R2,R3,R4);
Z1=measure(:,1)+n1;
Z2=measure(:,1)+n2;

X1init=zeros(size(A,1),1);
P1init=diag([Q Q]);
X2init=zeros(size(A,1),1);
P2init=diag([Q Q]);
Xinit=zeros(size(A,1),1);
Pinit=diag([Q Q]);
X(1)=0;

for i=2:samples
  %local filter 1
  [X1,P1,X1pred,P1pred]=central(A,B,[1 0],Q,u(i-1),R1,Z1(i),X1init,P1init);


  %local filter 2
  [X2,P2,X2pred,P2pred]=central(A,B,[1 0],Q,u(i-1),R2,Z2(i),X2init,P2init);
  X2init=X2;
  P2init=P2;


  Xloc=[X1; X2];
  Ploc=[P1; P2];
  Xlocpred=[X1pred; X2pred];
  Plocpred=[P1pred; P2pred];

  %fusion of local filters with hierred.m
  [Xglob,Pglob]=hierred(A,B,[1 0 ;1 0],Q,u(i-1),R2,Xinit,Pinit,Ploc,Plocpred,Xloc,Xlocpred);

  Xinit=Xglob;
  Pinit=Pglob;

  %feedback of global state estimate and global covariance to the local
  %estimators, according to equations 5.4 and 5.5
  X1init=Xglob;
  P1init=Pglob;

  X(i)=Xglob(1);
end

noise=X'-measure(:,1);
%mean(noise(100:samples))
%var(noise(100:samples))

subplot(2,1,1)
plot([1:500]*0.02,noise)
Title(['R1=',num2str(R1),' and R2=',num2str(R2)])
xlabel('t(sec)')
ylabel('noise (m)')
text(1,0.1,['mean noise: ',num2str(mean(noise(100:samples)))])
text(6,0.1,['var noise: ',num2str(var(noise(100:samples)))])

subplot(2,1,2)
plot([1:500]*0.02,P)
xlabel('t(sec)')
ylabel('P(1,1)')
```