Int. Journal of Math. Analysis, Vol. 3, 2009, no. 22, 1061 - 1080

Optimal Distributed Lainiotis Filter

Nicholas Assimakis

Department of Electronics
Technological Educational Institute (T.E.I.) of Lamia
35100 Lamia, Greece
assimakis@teilam.gr

Abstract

A method to implement the optimal distributed Lainiotis filter is proposed. The method is based on the a-priori determination of the optimal distribution of the measurements in parallel processors, in the sense of minimizing the computation time. The resulting optimal Lainiotis filter presents high parallelism speedup. This is very important due to the fact that, in most real-time applications, it is essential to obtain the estimate in the shortest possible time.

Mathematics Subject Classification: 93E11, 93C55, 68Q10, 68Q25

Keywords: Lainiotis filter, Discrete-time, Distributed algorithms, Analysis of algorithms

1 Introduction

Estimation plays an important role in many fields of science and engineering. The discrete time Lainiotis filter [6] is a well known algorithm that solves the estimation/filtering problem. Real time problems require fast and accurate computation of large amount of data in order to deal with larger and more realistic models. The advances in the technology of integrated multi-sensor network systems allow the use of distributed signal processing algorithms. A typical multi-sensor environment consists of several sensors observing a dynamic system, where each sensor is attached to a local processor; in these decentralized structures some amount of processing is done at the local processors of the network and the results are then communicated to a central processor, also referred to as a fusion center. In this paper the hierarchical approach for signal processing is used, in the case where the sensors are both collocated and dispersed [4]. The novelty of this paper for the important problem of distributed computing is the derivation of an optimal solution to the distributed problem: the optimal number of processors required for the parallel implementation of the Lainiotis filter, for which the maximum parallelism speedup is achieved,

is presented. It must be noted that the optimal allocation of the measurements into the set of the distributed processors can be obtained before the algorithm's implementation. It is also shown that the speedup is significant.

2 Problem Formulation

The estimation/filtering problem is associated with the following state space equations:

$$x(k+1) = F(k+1,k)x(k) + w(k) \tag{1}$$

$$z(k) = H(k)x(k) + v(k) \tag{2}$$

where $x(k)$ is the $n-$dimensional state vector at time k, $z(k)$ is the $m-$dimensional measurement vector, $F(k+1,k)$ is the $n \times n$ system transition matrix, $H(k)$ is the $m \times n$ output matrix, $\{w(k)\}$ is the plant noise processes and $\{v(k)\}$ is the measurement noise process. These processes are assumed to be Gaussian, zero-mean, white and uncorrelated random processes with plant noise and measurement noise covariance matrices $Q(k)$ and $R(k)$, respectively. The vector $x(0)$ is a Gaussian random process with mean x_0 and covariance P_0. Also, $x(0)$, $\{w(k)\}$ and $\{v(k)\}$ are independent.
For a multi-sensor environment, the measurement vector is partitioned into P parts [4], [9]:

$$z^T(k) = \left[\begin{array}{cccc} z_1^T(k) & z_2^T(k) & \dots & z_P^T(k) \end{array} \right] \tag{3}$$

where

$$\sum_{i=1}^{P} m_i = m \tag{4}$$

Partitioning the measurement noise process into P statistically uncorrelated parts as:

$$v^T(k) = \left[\begin{array}{cccc} v_1^T(k) & v_2^T(k) & \dots & v_P^T(k) \end{array} \right] \tag{5}$$

where

$$E\{v(k)v^T(k)\} = diag \left(\begin{array}{cccc} R_1(k), & R_2(k), & \dots, & R_P(k) \end{array} \right) \tag{6}$$

and partitioning the output matrix as:

$$H^T(k) = \left[\begin{array}{cccc} H_1^T(k) & H_2^T(k) & \dots & H_P^T(k) \end{array} \right] \tag{7}$$

the measurement equation takes the form:

$$z_i(k) = H_i(k)x(k) + v_i(k), \quad i = 1, 2, \dots, p \tag{8}$$

3 Centralized Lainiotis Filter

In this section we present the discrete time Lainiotis Filter [6] in a centralized form. This means that all the required computations are carried out in one central processor. For time varying systems, the classical implementation of the Lainiotis filter is summarized in the following equations:

LF (Lainiotis Filter)

$$A(k+1) = \left[H(k+1)Q(k)H^T(k+1) + R(k+1)\right]^{-1} \tag{9}$$

$$K_n(k+1) = Q(k)H^T(k+1)A(k+1) \tag{10}$$

$$K_m(k+1) = F^T(k+1,k)H^T(k+1)A(k+1) \tag{11}$$

$$P_n(k+1,k) = Q(k) - K_n(k+1)H(k+1)Q(k) \tag{12}$$

$$F_n(k+1,k) = F(k+1,k) - K_n(k+1)H(k+1)F(k+1,k) \tag{13}$$

$$O_n(k+1) = K_m(k+1)H(k+1)F(k+1,k) \tag{14}$$

$$x_n(k+1/k+1) = K_n(k+1)z(k+1) \tag{15}$$

$$M_n(k+1/k+1) = K_m(k+1)z(k+1) \tag{16}$$

$$\begin{aligned} P(k+1/k+1) &= P_n(k+1,k) \\ &\quad + F_n(k+1,k)\left[I + P(k/k)O_n(k+1)\right]^{-1} P(k/k)F_n^T(k+1,k) \end{aligned} \tag{17}$$

$$\begin{aligned} x(k+1/k+1) &= x_n(k+1,k) \\ &\quad + F_n(k+1,k)\left[I + P(k/k)O_n(k+1)\right]^{-1}\left[P(k/k)M_n(k+1) + x(k/k)\right] \end{aligned} \tag{18}$$

for $k = 1, 2, \ldots$, with initial conditions $x(0/0) = x_0$ and $P(0/0) = P_0$.

After some algebra, the following implementation of the Centralized Time Varying Lainiotis Filter is derived, as shown in Appendix A:

CTVLF (Centralized Time Varying Lainiotis Filter)

$$P_n(k+1,k) = \left[Q^{-1}(k) + H^T(k+1)R^{-1}(k+1)H(k+1)\right]^{-1} \tag{19}$$

$$F_n(k+1,k) = P_n(k+1,k)Q^{-1}(k)F(k+1,k) \tag{20}$$

$$O_n(k+1) = \left[Q^{-1}(k)F(k+1,k)\right]^T \left[Q(k) - P_n(k+1,k)\right]\left[Q^{-1}(k)F(k+1,k)\right] \tag{21}$$

$$\begin{aligned} P(k+1/k+1) &= P_n(k+1,k) \\ &\quad + F_n(k+1,k)\left[I + P(k/k)O_n(k+1)\right]^{-1} P(k/k)F_n^T(k+1,k) \end{aligned} \tag{22}$$

$$\begin{aligned} x(k+1/k+1) &= F_n(k+1,k)\left[I + P(k/k)O_n(k+1)\right]^{-1} x(k/k) \\ &\quad + P(k+1/k+1)H^T(k+1)R^{-1}(k+1)z(k+1) \end{aligned} \tag{23}$$

For time invariant systems, the system transition matrix, the output matrix, the plant and measurement noise covariance matrices are constant. Then, the Centralized Time Invariant Lainiotis Filter is obtained:

CTILF (Centralized Time Invariant Lainiotis Filter)

$$P(k+1/k+1) = P_n + F_n \left[I + P(k/k)O_n\right]^{-1} P(k/k)F_n^T \tag{24}$$

$$x(k+1/k+1) = F_n \left[I + P(k/k)O_n\right]^{-1} x(k/k) + P(k+1/k+1)H^T R^{-1} z(k+1) \tag{25}$$

where the following constant matrices are calculated off-line:

$$P_n = \left[Q^{-1} + H^T R^{-1} H\right]^{-1} \tag{26}$$

$$F_n = P_n Q^{-1} F \tag{27}$$

$$O_n = \left[Q^{-1}F\right]^T \left[Q - P_n\right] \left[Q^{-1}F\right] \tag{28}$$

For time invariant systems, it is well known [1] that if the signal process model is asymptotically stable (i.e. all eigenvalues of F lie inside the unit circle), then there exist a steady state value $\overline{P}$ of the estimation error covariance matrix. Then, the Centralized Steady State Lainiotis Filter is obtained:

CSSLF (Centralized Steady State Lainiotis Filter)

$$x(k+1/k+1) = A_{ss}x(k/k) + B_{ss}z(k+1) \tag{29}$$

where the following constant matrices are calculated off-line:

$$A_{ss} = F_n \left[I + \overline{P}O_n\right]^{-1} \tag{30}$$

$$B_{ss} = \overline{P}H^T R^{-1} \tag{31}$$

after having calculated off-line the constant matrices F_n, O_n using (26)-(28) and having calculated the steady state estimation error covariance matrix $\overline{P}$ by off-line solving the corresponding discrete time Riccati equation [3], [5]:

$$\overline{P} = P_n + F_n \left[I + \overline{P}O_n\right]^{-1} \overline{P}F_n^T \tag{32}$$

4 Distributed Lainiotis Filter

In this section we present the discrete time Lainiotis in a distributed form [4]. As was pointed out earlier, the main drawback of the above centralized approaches is that they require a large amount of computations to be carried out in the central processor, demanding therefore large computational power. Moreover in the case of very large m , there is a tremendous computational burden in the processor.

In the following, the results of the previous section are extended and the corresponding distributed algorithms are decomposed into two parts: the Local Level and the Central Level. At the Local Level each processor computes its local estimate using its own measurement. The data of each local processor is communicated to the fusion center where the global estimate is computed. The local processors can operate concurrently, since there is no need for communication among local processors and no communication is needed from

the central processor downwards in the hierarchy of the local processors. The generation of the global estimate can be thought of as overhead, due to the fact that the central processor needs information from the local processors, but not vice-versa.

For time varying systems, the following implementation of the Distributed Time Varying Lainiotis Filter is derived:

DTVLF (Distributed Time Varying Lainiotis Filter)

Local Level

$$B_i(k+1) = H_i^T(k+1)R_i^{-1}(k+1)H_i(k+1),\ i = 1, \ldots, P \tag{33}$$

$$b_i(k+1) = H_i^T(k+1)R_i^{-1}(k+1)z_i(k+1),\ i = 1, \ldots, P \tag{34}$$

Central Level

$$P_n(k+1,k) = \left[Q^{-1}(k) + \sum_{i=1}^{P} B_i(k+1)\right]^{-1} \tag{35}$$

$$F_n(k+1,k) = P_n(k+1,k)Q^{-1}(k)F(k+1,k) \tag{36}$$

$$O_n(k+1) = \left[Q^{-1}(k)F(k+1,k)\right]^T \left[Q(k) - P_n(k+1,k)\right] \left[Q^{-1}(k)F(k+1,k)\right] \tag{37}$$

$$\begin{aligned} P(k+1/k+1) &= P_n(k+1,k) \\ &+F_n(k+1,k)\left[I + P(k/k)O_n(k+1)\right]^{-1} P(k/k)F_n^T(k+1,k) \end{aligned} \tag{38}$$

$$\begin{aligned} x(k+1/k+1) &= F_n(k+1,k)\left[I + P(k/k)O_n(k+1)\right]^{-1} x(k/k) \\ &+P(k+1/k+1)\sum_{i=1}^{P} b_i(k+1) \end{aligned} \tag{39}$$

For time invariant systems, using the equations of the Distributed Time Invariant Lainiotis Filter, we obtain the Distributed Time Invariant Lainiotis Filter:

DTILF (Distributed Time Invariant Lainiotis Filter)

Local Level

$$b_i(k+1) = H_i^T R_i^{-1} z_i(k+1),\ i = 1, \ldots, P \tag{40}$$

where the constant matrices $H_i^T R_i^{-1},\ i = 1, \ldots, p$ are calculated off-line.

Central Level

$$P(k+1/k+1) = P_n + F_n\left[I + P(k/k)O_n\right]^{-1} P(k/k)F_n^T \tag{41}$$

$$x(k+1/k+1) = F_n\left[I + P(k/k)O_n\right]^{-1} x(k/k) + P(k+1/k+1)\sum_{i=1}^{P} b_i(k+1) \tag{42}$$

where the constant matrices $P_n\,,\ F_n\,,\ O_n$ are calculated off-line using (26)-(28).

In the steady state case, using the equations of the Distributed Steady State Lainiotis

Filter, we obtain the Distributed Steady State Lainiotis Filter:

DSSLF (Distributed Steady State Lainiotis Filter)

Local Level

$$d_i(k+1) = \overline{P} H_i^T R_i^{-1} z_i(k+1),\ i = 1, \ldots, P \tag{43}$$

where the constant matrices $\overline{P} H_i^T R_i^{-1}, i = 1, \ldots, p$ are calculated off-line after having calculated the steady state estimation error covariance matrix $\overline{P}$ by off-line solving the corresponding discrete time Riccati equation [3], [5].

Central Level

$$x(k+1/k+1) = A_{ss} x(k/k) + \sum_{i=1}^{P} d_i(k+1) \tag{44}$$

where the constant matrix A_{ss} is calculated off-line using (12), after having calculated off-line the constant matrices F_n, O_n using (26)-(28) and having calculated the steady state estimation error covariance matrix $\overline{P}$ by off-line solving the corresponding discrete time Riccati equation.

5 Computational Requirements

Both the centralized and the distributed Lainiotis filters are recursive algorithms. Thus, the total computational time required for the implementation of each centralized algorithm is equal to:

$$t_C(alg) = CB_C(alg)s(alg)t_{op} \tag{45}$$

while the total computational time required for the implementation of each distributed algorithm is equal to:

$$t_D(alg) = CB_D(alg)s(alg)t_{op} \tag{46}$$

where $CB_C(alg)$ and $CB_D(alg)$ are the per recursion calculation burdens required for the on-line calculations of each centralized and distributed algorithm respectively, $s(alg)$ is the number of recursions that each algorithm executes and t_{op} is the time required to perform a scalar operation.

The centralized and the distributed algorithms presented above are equivalent with respect to their behavior: they calculate theoretically the same estimates. Then, it is reasonable to assume that both implementations compute the estimate value $x(L/L)$ of the state vector $x(L)$, executing the same number of recursions. Thus, in order to compare the algorithms with respect to their computational time , we have to compare their per recursion calculation burden required for the on-line calculations; the calculation burden of the off-line calculations (initialization process for time invariant and steady state filters) is not taken into account.

The computational analysis is based on the analysis in [2]: scalar operations are involved in matrix manipulation operations, which are needed for the implementation of the filtering algorithms. Table 1 summarizes the calculation burden of needed matrix operations.

Table 1. Calculation burden of matrix operations

Matrix Operation	Calculation Burden
$A(nxm) + B(nxm) = C(nxn)$	nm
$A(nxn) + B(nxn) = S(nxn)\,[S : symmetric]$	$0.5n^2 + 0.5n$
$I(nxn) + A(nxn) = B(nxn)\,[I : identity]$	n
$A(nxm) \cdot B(mxk) = C(nxk)$	$2nmk - nk$
$A(nxm) \cdot B(mxn) = S(nxn)\,[S : symmetric]$	$n^2m + nm - 0.5n^2 - 05n$
$[A(nxm)]^{-1} = B(nxm)$	$(16n^3 - 3n^2 - n)/6$

In the centralized algorithms case, all required computations for the calculation of the global estimate are carried out in one central processor. The computational requirements of all centralized algorithms depend on the state vector dimension n and the measurement vector dimension m.

In the distributed algorithms case, each local processor computes its local data using its

own measurement (the local processors operate in parallel, since there is no need for communication among local processors). The data of each local processor is communicated to the central processor (there is no two way communication between the local and the central processor), where the global estimate is computed. The computational requirements of all distributed algorithms depend on the state vector dimension n and the maximum local measurement vector dimension M:

$$M = \max\{m_i\},\ i = 1, \ldots, p \tag{47}$$

The per recursion calculation burdens of all the Lainiotis filtering algorithms are determined in Appendix B and summarized in Table 2. It is obvious that when $P = 1$ and $M = m$, then the distributed algorithms become the same as the centralized algorithms with equivalent calculation burdens.

Table 2. Computational Requirements
Per recursion calculation burden of the Lainiotis filtering algorithms

System	Algorithm	Calculation Burden
Centralized Time Varying	CTVLF	$(44n^3 - 3n^2 - 3n)/2$ $n^2m + 2nm + 2nm^2$ $(16m^3 - 3m^2 - n)/6$
Centralized Time Invariant	CTILF	$(58n^3 + 9n^2 - 7n)/6 + 2nm$
Centralized Steady State	CSSLF	$2n^2 + 2nm - n$
Distributed Time Varying	DTVLF	$(22n^3 - 2n^2 - 3n)$ $n^2M + 2nM + 2nM^2$ $(16M^3 - 3M^2 - M)/6 + P(0.5n^2 + 1.5n)$
Distributed Time Invariant	DTILF	$(58n^3 + 9n^2 - 13n)/6 + Pn + 2nM$
Distributed Steady State	DSSLF	$2n^2 - 2n + Pn + 2nM$

6 Optimal Distributed Lainotis Filter Definition

In this section, a method to define the optimal distributed Lainiotis filter is proposed. Our aim is to minimize the computation time of the distributed filter. From the previous section, it becomes obvious that in order to minimize the computation time of the distributed algorithm, we have to minimize the corresponding per recursion calculation burden. From Table 2 we can easily see that the calculation burden of the distributed Lainiotis filter depends on the distribution of m measurements into P parallel local processors. Each local processor can transfer data to the central processor after its local data is computed in the local processor; this communication process can be performed while the other local processors compute their local estimates. The global estimate is computed in the central processor after all needed information is transferred from the local processors to the central processor. We make the following assumption:

The measurement vector is partitioned into P equal parts. Then, it is clear that the following relation hold:

$$m_i = M,\ i = 1, \ldots, P \tag{48}$$

From (4) we have:

$$P \cdot M = m \tag{49}$$

This uniform distribution of measurements into processors has the following advantages:

1. All local processors perform the same calculations concerning quantities of the same type and dimensionality. Thus, all local processors have the same structure and therefore, low hardware cost is required for the implementation of the distributed algorithms.

2. There is no idle time for the local processors.

In order to determine the Optimal Distributed Lainiotis Filter, we observe that P and M must be positive integers satisfying (48). From Table 2 using (47)-(48) we conclude that we are able to determine the optimal uniform distribution of m measurements into P_{opt} processors, each one of which deals with M_{opt} measurements, using the optimality criterion of minimizing the computation time of the distributed Lainiotis filter.
For time varying systems, the calculation burden of the distributed Lainiotis filter becomes minimum when the following quantity is minimum:

$$q_{TVLF} = n^2M + 2nM + 2nM^2 + (16M^3 - 3M^2 - M)/6 + P(0.5n^2 + 1.5n) \tag{50}$$

Computing the derivative with respect to M or P , this quantity has minimum when:

$$\left(\begin{array}{c} 8M^4 + (4n-1)M^3 + (n^2 + 2n - 1/6)M^2 - (n(n+3)m/2) = 0, \\ \text{or} \\ -(n(n+3)/2)P^4 + (2n^2 + 2n - 1/6)mP^2 + (4n-1)m^2P + 8m^3 = 0 \end{array} \right) \tag{51}$$

For time invariant systems or steady state systems, the calculation burden of the distributed Lainiotis filter becomes minimum when the following quantity is minimum:

$$q_{TILF,SSLF} = 2M + P \tag{52}$$

Computing the derivative with respect to M or P , this quantity has minimum when:

$$\begin{pmatrix} M^2 = m/2 \\ \text{or} \\ P^2 = 2m \end{pmatrix} \tag{53}$$

Then, the optimum distribution is determined by solving (51) for time varying systems or (53) for time invariant systems or steady state systems, with P and M positive integers; the optimum distribution consists of M_{opt} measurements in each of P_{opt} processors.

The parallelism speedup of each distributed Lainiotis algorithm is defined as the computational time required for its centralized implementation divided by the computational time required for its distributed implementation:

$$speedup(alg) = \frac{t_C(alg)}{t_D(alg)} = \frac{CB_C(alg)s(alg)t_{op}}{CB_D(alg)s(alg)t_{op}} = \frac{CB_C(alg)}{CB_D(alg)} \tag{54}$$

Thus, the optimal distributed algorithms present the maximum parallelism speedup,which is achieved minimizing the computational time required for the algorithm's distributed implementation. The maximum parallelism speedup increases as the measurement vector dimension increases (and the state vector dimension remains constant), for time varying, time invariant and steady state systems. This is very important for multi-sensor problems.

7 Simulation Results

In this section it is pointed out that the proposed Optimal Distributed Lainiotis Filter presents high parallelism speedup.

Example 1. Time Varying Lainiotis filter.

A typical multi-sensor example taken from [8] is presented. A time varying system with $n = 1$ and $m = 1000$ is considered.

The Optimal Distributed Lainiotis Filter is achieved for the optimum uniform distribution:

$M_{opt} = 4$ measurements in each of $P_{opt} = 250$ local processors

The parallelism speedup with respect to the number of processors required for the implementation of the time varying distributed Lainiotis filter, is plotted in Figure 1.

The Optimal Distributed Lainiotis Filter presents a very high maximum parallelism speedup: the Optimal Distributed Lainiotis Filter can be implemented $3.7 \cdot 10^6$ times faster than the Centralized Lainiotis Filter.

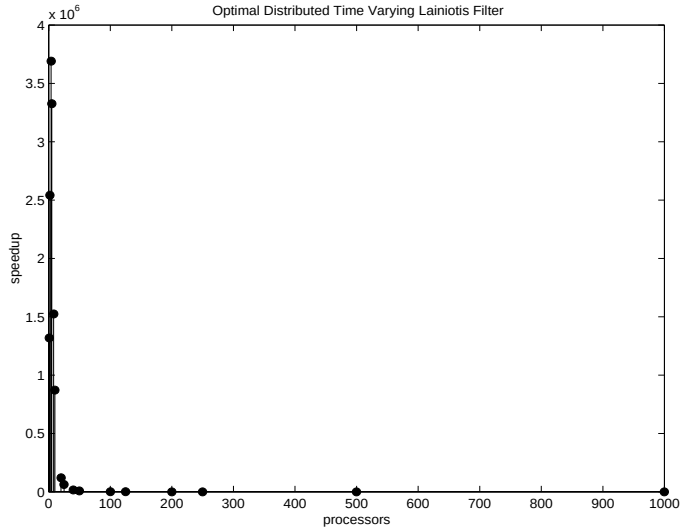

Figure 1: Parallelism speedup for time varying systems ($n = 1$ and $m = 1000$)

Example 2. Time Invariant Lainiotis filter.

A typical seismic deconvolution example taken from [7] is presented. The time invariant wavelet used to describe the signal received by the seismic sensors is assumed of order $n = 4$. A number of $m = 1000$ sensors divided into P local geophone clusters are utilized in order to capture the seismic trace.

The Optimal Distributed Lainiotis Filter arises for two different uniform distributions:

$M_{opt} = 25$ measurements in each of $P_{opt} = 40$ local processors

or

$M_{opt} = 20$ measurements in each of $P_{opt} = 50$ local processors

In this case, we propose to determine the optimum distribution using the criterion of

minimizing the number of processors (except the criterion of minimizing the computation time). This is reasonable since the hardware cost is minimized.
The parallelism speedup with respect to the number of processors required for the implementation of the time invariant distributed Lainiotis filter, is plotted in Figure 2.
The Optimal Distributed Lainiotis Filter can be implemented 8.7 times faster than the Centralized Lainiotis Filter. This is achieved for both the above distributions.

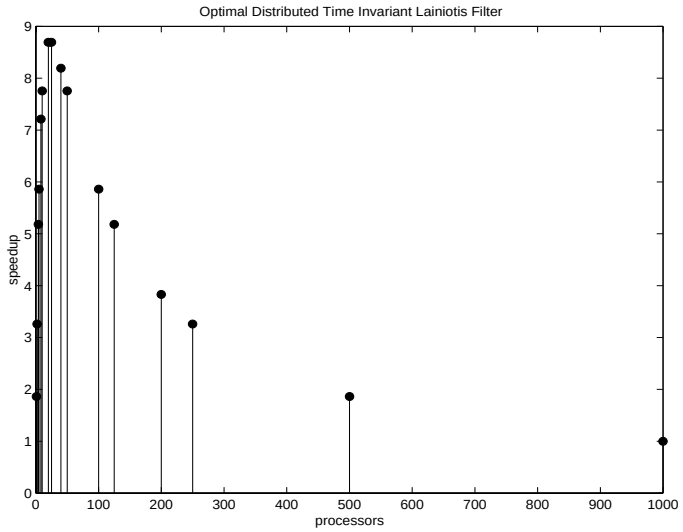

Figure 2: Parallelism speedup for time invariant systems ($n = 4$ and $m = 1000$)

8 Conclusions

Centralized and distributed algorithms for the solution of the discrete time estimation/filtering problem for multi-sensor environment were presented. The discrete time Centralized and Distributed Lainiotis Filters were analyzed for time varying, time invariant and steady state systems and their computational requirements were discussed. A method was developed to define the Optimal Distributed Lainiotis Filter a-priori, i.e. before the implementation of the filter. The method is based on the determination of the optimum uniform distribution of measurements into parallel processors using the criterion of minimizing the computation time. It was verified through simulation results that the proposed Optimal Distributed Lainiotis Filter presents high parallelism speedup. This result is very important due to the fact that, in most real-time applications, it is essential to obtain the estimate in the shortest possible time.

References

[1] B.D.O. Anderson and J.B. Moore, Optimal Filtering, Prentice Hall inc., 1979.

[2] N. Assimakis, M. Adam, Discrete time Kalman and Lainiotis filters comparison, Int. Journal of Mathematical Analysis (IJMA), **1** (2007), 635–659.

[3] N. Assimakis, D. Lainiotis, S. Katsikas and F. Sanida, A Survey of Recursive Algorithms for the Solution of the Discrete Time Riccati Equation, Nonlinear Analysis, Theory, Methods and Applications, **30** (1977), 2409–2420.

[4] H.R. Hashemipour, S. Roy and A.J. Laub, Decentralized structures for parallel Kalman filtering, IEEE Trans. on Automatic Control, **33** (1988), 88–94.

[5] D.G. Lainiotis, Discrete Riccati Equation Solutions: Partitioned Algorithms, IEEE Transactions on AC, **AC-20** (1975), 555–556.

[6] D.G. Lainiotis, Partitioned linear estimation algorithms: Discrete case, IEEE Trans. on AC, **AC-20** (1975), 255–257.

[7] D.G. Lainiotis, S.K. Katsikas and S.D. Likothanassis,Optimal seismic deconvolution, Signal Processing, **15** (1988), 375–404.

[8] D.G. Lainiotis, K.N. Plataniotis, M. Papanikolaou, P. Papaparaskeva P., Discrete Riccati Equation Solutions - Distributed Algorithms, Mathematical Problems in Engineering, **2** (1996), 319–322.

[9] K.N. Plataniotis, D.G. Lainiotis, Multisensor Estimation - New Distributed Algorithms, Mathematical Problems in Engineering, **3** (1996), 27–52.

Appendix A. Centralized Time Varying Lainiotis Filter (CTVLF)

From the Lainiotis Filter (LF) equations (9)-(18) are going to derive the Centralized Time Varying Lainiotis Filter (CTVLF) equations (19)-(23).

First we focus on equations (19)-(21).

From (12) and (13) we have:

$$[I - K_n(k+1)H(k+1)] = P_n(k+1,k)Q^{-1}(k) = F_n(k+1,k)F^{-1}(k+1,k)$$

from where we derive (20):

$$F_n(k+1,k) = P_n(k+1,k)Q^{-1}(k)F(k+1,k) \tag{A.1}$$

From (2) and (4) we have:

$$P_n(k+1,k) = Q(k) - Q(k)H^T(k+1)A(k+1)H(k+1)Q(k) \tag{A.2}$$

and from (3) and (6) we have:

$$O_n(k+1) = F^T(k+1,k)H^T(k+1)A(k+1)H(k+1)F(k+1,k) \tag{A.3}$$

From (A.1) and (A.3) we have:

$$\begin{aligned} O_n(k+1) = {} & F_n^T(k+1,k)P_n^{-1}(k+1,k) \\ & Q(k)H^T(k+1)A(k+1)H(k+1)Q(k) \\ & P_n^{-1}(k+1,k)F_n(k+1,k) \end{aligned} \tag{A.4}$$

Then, from (A.4) and using (A.1) and (A.2) we derive (21):

$$O_n(k+1) = \left[Q^{-1}(k)F(k+1,k)\right]^T \left[Q(k) - P_n(k+1,k)\right] \left[Q^{-1}(k)F(k+1,k)\right] \tag{A.5}$$

From (9) and (A.2) we have:

$$\begin{aligned} P_n(k+1,k) = {} & Q(k) \\ & -Q(k)H^T(k+1)[H(k+1)Q(k)H^T(k+1) + R(k+1)]^{-1}H(k+1)Q(k) \end{aligned}$$

Then,using the matrix inversion lemma:

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[C^{-1} + DA^{-1}B]^{-1}DA^{-1} \tag{A.6}$$

we derive (19):

$$P_n(k+1,k) = \left[Q^{-1}(k) + H^T(k+1)R^{-1}(k+1)H(k+1)\right]^{-1} \tag{A.7}$$

In the sequel we focus on equations (22)-(23).

It is obvious that equation (17) coincides with equation (22).

Furthermore, from (15), (16) and (18) we have:

$$\begin{aligned} x(k+1/k+1) = {} & F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}x(k/k) \\ & +[K_n(k+1) + F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}P(k/k)K_m(k+1)]z(k+1) \end{aligned}$$

Then, using (10) and (11) we have:

$$\begin{aligned} x(k+1/k+1) = {} & F_n(k+1,k)[I+P(k/k)O_n(k+1)]^{-1}x(k/k) \\ & +Q(k)H^T(k+1)A(k+1)z(k+1) \\ & +F_n(k+1,k)[I+P(k/k)O_n(k+1)]^{-1}P(k/k) \\ & F^T(k+1,k)H^T(k+1)A(k+1)z(k+1) \end{aligned} \tag{A.8}$$

From (22) and (23) we have:

$$\begin{aligned} x(k+1/k+1) = {} & F_n(k+1,k)\left[I+P(k/k)O_n(k+1)\right]^{-1}x(k/k) \\ & +P(k+1/k+1)H^T(k+1)R^{-1}(k+1)z(k+1) \\ & = F_n(k+1,k)[I+P(k/k)O_n(k+1)]^{-1}x(k/k) \\ & +P_n(k+1,k)H^T(k+1)R^{-1}(k+1)z(k+1) \\ & +F_n(k+1,k)\left[I+P(k/k)O_n(k+1)\right]^{-1}P(k/k)F_n^T(k+1,k) \\ & H^T(k+1)R^{-1}(k+1)z(k+1) \end{aligned}$$

Then, using (A.1) we have:

$$\begin{aligned} & x(k+1/k+1) = F_n(k+1,k)[I+P(k/k)O_n(k+1)]^{-1}x(k/k) \\ & +P_n(k+1,k)H^T(k+1)R^{-1}(k+1)z(k+1) \\ & +F_n(k+1,k)\left[I+P(k/k)O_n(k+1)\right]^{-1}P(k/k)F^T(k+1,k)Q^{-1}(k)P_n(k+1,k) \\ & H^T(k+1)R^{-1}(k+1)z(k+1) \end{aligned} \tag{A.9}$$

Finally, from (4) and (A.2) we have:

$$\begin{aligned} & A(k+1) = \left[H(k+1)Q(k)H^T(k+1)+R(k+1)\right]^{-1} \\ & \Rightarrow A(k+1)[H(k+1)Q(k)H^T(k+1)+R(k+1)] = I \\ & \Rightarrow Q(k)H^T(k+1)A(k+1)[H(k+1)Q(k)H^T(k+1)+R(k+1)] = Q(k)H^T(k+1) \\ & \Rightarrow Q(k)H^T(k+1)A(k+1)H(k+1)Q(k)H^T(k+1) \\ & \quad +Q(k)H^T(k+1)A(k+1)R(k+1) = Q(k)H^T(k+1) \\ & \Rightarrow [Q(k)-P_n(k+1,k)]H^T(k+1) \\ & \quad +Q(k)H^T(k+1)A(k+1)R(k+1) = Q(k)H^T(k+1) \\ & \Rightarrow Q(k)H^T(k+1)A(k+1)R(k+1) = P_n(k+1,k)H^T(k+1) \end{aligned}$$

from where we derive:

$$Q(k)H^T(k+1)A(k+1) = P_n(k+1,k)H^T(k+1)R^{-1}(k+1) \tag{A.10}$$

Then, from (A.10), it is clear that (A.8) is equivalent to (A.9).
Thus, equations (19)-(23) are derived from equations (9)-(18).

Appendix B. Per recursion calculation burdens of Lainiotis filters

Matrix Operation	Matrix Dimensions	Calculation Burden
$Q^{-1}(k)$	$(n \times n)^{-1}$	$(16n^3 - 3n^2 - n)/6$
$R^{-1}(k+1)$	$(m \times m)^{-1}$	$(16m^3 - 3m^2 - m)/6$
$H^T(k+1)R^{-1}(k+1)$	$(n \times m) \cdot (m \times m)$	$2nm^2 - nm$
$H^T(k+1)R^{-1}(k+1)H(k+1)$	$(n \times n) \cdot (n \times n)$ *symmetric*	$n^2m + nm - 0.5n^2 - 0.5n$
$Q^{-1}(k) + H^T(k+1)R^{-1}(k+1)H(k+1)$	$(n \times n) + (n \times n)$ *symmetric*	$0.5n^2 + 0.5n$
$P_n(k+1,k)$ $= [Q^{-1}(k) + H^T(k+1)R^{-1}(k+1)H(k+1)]^{-1}$	$(n \times n)^{-1}$	$(16n^3 - 3n^2 - n)/6$
$Q^{-1}(k)F(k+1,k)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n(k+1,k) = P_n(k+1,k)Q^{-1}(k)F(k+1,k)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$Q(k) - P_n(k+1,k)$	$(n \times n) + (n \times n)$ *symmetric*	$0.5n^2 + 0.5n$
$[Q(k) - P_n(k+1,k)][Q^{-1}(k)F(k+1,k)]$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$O_n(k+1) = [Q^{-1}(k)F(k+1,k)]^T$ $[Q(k) - P_n(k+1,k)][Q^{-1}(k)F(k+1,k)]$	$(n \times n) \cdot (n \times n)$ *symmetric*	$n^3 + 0.5n^2 - 0.5n$
$P(k/k)O_n(k+1)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$I + P(k/k)O_n(k+1)$	$I(n \times n) + (n \times n)$	n
$[I + P(k/k)O_n(k+1)]^{-1}$	$(n \times n)^{-1}$	$(16n^3 - 3n^2 - n)/6$
$F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}P(k/k)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}P(k/k)$ $F_n^T(k+1,k)$	$(n \times n) \cdot (n \times n)$ *symmetric*	$n^3 + 0.5n^2 - 0.5n$
$P(k+1/k+1) = P_n(k+1,k)$ $+F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}P(k/k)$ $F_n^T(k+1,k)$	$(n \times n) + (n \times n)$ *symmetric*	$0.5n^2 + 0.5n$
$F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}x(k/k)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$[H^T(k+1)R^{-1}(k+1)]z(k+1)$	$(n \times m) \cdot (m \times 1)$	$2nm - n$
$P(k+1/k+1)H^T(k+1)R^{-1}(k+1)z(k+1)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$x(k+1/k+1)$ $= F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}x(k/k)$ $+P(k+1/k+1)H^T(k+1)R^{-1}(k+1)z(k+1)$	$(n \times 1) + (n \times 1)$	n
	Total	$(44n^3 - 3n^2 - 3n)/2$ $+n^2m + 2nm + 2nm^2$ $+(16m^3 - 3m^2 - m)/6$

Centralized Time Invariant Lainiotis Filter (CTILF)

Matrix Operation	Matrix Dimensions	Calculation Burden
$P(k/k)O_n$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$I + P(k/k)O_n$	$I(n \times n) + (n \times n)$	n
$[I + P(k/k)O_n]^{-1}$	$(n \times n)^{-1}$	$(16n^3 - 3n^2 - n)/6$
$F_n[I + P(k/k)O_n]^{-1}$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n[I + P(k/k)O_n]^{-1}P(k/k)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n[I + P(k/k)O_n]^{-1}P(k/k)F_n^T$	$(n \times n) \cdot (n \times n)$ *symmetric*	$n^3 + 0.5n^2 - 0.5n$
$P(k+1/k+1) = P_n$ $+F_n[I + P(k/k)O_n]^{-1}P(k/k)F_n^T$	$(n \times n) + (n \times n)$ *symmetric*	$0.5n^2 + 0.5n$
$F_n[I + P(k/k)O_n]^{-1}x(k/k)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$[H^T R^{-1}]z(k+1)$	$(n \times m) \cdot (m \times 1)$	$2nm - n$
$P(k+1/k+1)H^T R^{-1}z(k+1)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$x(k+1/k+1) = F_n[I + P(k/k)O_n]^{-1}x(k/k)$ $+P(k+1/k+1)H^T R^{-1}z(k+1)$	$(n \times 1) + (n \times 1)$	n
	Total	$(58n^3 + 9n^2 - 7n)/6$ $+2nm$

Centralized Steady State Lainiotis Filter (CSSLF)

Matrix Operation	Matrix Dimensions	Calculation Burden
$A_{ss}x(k/k)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$B_{ss}z(k+1)$	$(n \times m) \cdot (m \times 1)$	$2nm - n$
$x(k+1/k+1) = A_{ss}x(k/k) + B_{ss}z(k+1)$	$(n \times 1) + (n \times 1)$	n
	Total	$2n^2 + 2nm - n$

Distributed Time Varying Lainiotis Filter (DTVLF)

Central Level

Matrix Operation	Matrix Dimensions	Calculation Burden
$Q^{-1}(k)$	$(n \times n)^{-1}$	$(16n^3 - 3n^2 - n)/6$
$\sum_{i=1}^{P} B_i(k+1)$	$(n \times n) \cdot (n \times n)$ *symmetric*	$(P-1)(0.5n^2 + 0.5n)$
$Q^{-1}(k) + \sum_{i=1}^{P} B_i(k+1)$	$(n \times n) + (n \times n)$ *symmetric*	$0.5n^2 + 0.5n$
$P_n(k+1,k) = [Q^{-1}(k) + \sum_{i=1}^{P} B_i(k+1)]^{-1}$	$(n \times n)^{-1}$	$(16n^3 - 3n^2 - n)/6$
$Q^{-1}(k)F(k+1,k)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n(k+1,k) = P_n(k+1,k)Q^{-1}(k)F(k+1,k)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$Q(k) - P_n(k+1,k)$	$(n \times n) + (n \times n)$ *symmetric*	$0.5n^2 + 0.5n$
$[Q(k) - P_n(k+1,k)][Q^{-1}(k)F(k+1,k)]$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$O_n(k+1) = [Q^{-1}(k)F(k+1,k)]^T$ $[Q(k) - P_n(k+1,k)][Q^{-1}(k)F(k+1,k)]$	$(n \times n) \cdot (n \times n)$ *symmetric*	$n^3 + 0.5n^2 - 0.5n$
$P(k/k)O_n(k+1)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$I + P(k/k)O_n(k+1)$	$I(n \times n) + (n \times n)$	n
$[I + P(k/k)O_n(k+1)]^{-1}$	$(n \times n)^{-1}$	$(16n^3 - 3n^2 - n)/6$
$F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}P(k/k)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}P(k/k)$ $F_n^T(k+1,k)$	$(n \times n) \cdot (n \times n)$ *symmetric*	$n^3 + 0.5n^2 - 0.5n$
$P(k+1/k+1) = P_n(k+1,k)$ $+F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}P(k/k)$ $F_n^T(k+1)$	$(n \times n) + (n \times n)$ *symmetric*	$0.5n^2 + 0.5n$
$F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}x(k/k)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\sum_{i=1}^{P} b_i(k+1)$	$(n \times n) \cdot (n \times 1)$	$(P-1)n$
$P(k+1/k+1)\sum_{i=1}^{P} b_i(k+1)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$x(k+1/k+1)$ $= F_n(k+1,k)[I + P(k/k)O_n(k+1)]^{-1}x(k/k)$ $+P(k+1/k+1)\sum_{i=1}^{P} b_i(k+1)$	$(n \times 1) + (n \times 1)$	n
	Total Central Level	$(44n^3 - 3n^2 - 3n)/2$ $+P(0.5n^2 + 1.5n)$

Local Level

Matrix Operation	Matrix Dimensions	Calculation Burden
$R_i^{-1}(k+1)$	$(M \times M)^{-1}$	$(16M^3 - 3M^2 - M)/6$
$H_i^T(k+1)R_i^{-1}(k+1)$	$(n \times M) \cdot (M \times M)$	$2nM^2 - nM$
$B_i(k+1) = H_i^T(k+1)R_i^{-1}(k+1)H_i(k+1)$	$(n \times M) \cdot (M \times n)$ *symmetric*	$n^2M + nM - 0.5n^2 - 0.5n$
$b_i(k+1) = H_i^T(k+1)R_i^{-1}(k+1)z_i(k+1)$	$(n \times M) \cdot (M \times 1)$	$2nM - n$
	Total Local Level	$n^2M + 2nM + 2nM^2$ $+(16M^3 - 3M^2 - M)/6$ $-(0.5n^2 + 1.5n)$

Level	Calculation Burden
Central Level	$(44n^3 - 3n^2 - 3n)/2 + P(0.5n^2 + 1.5n)$
Local Level	$n^2M + 2nM + 2nM^2 + (16M^3 - 3M^2 - M)/6 - (0.5n^2 + 0.5n)$
Total	$(22n^3 - 2n^2 - 3n)$ $+n^2M + 2nM + 2nM^2 + (16M^3 - 3M^2 - M)/6 + P(0.5n^2 + 1.5n)$

Distributed Time Invariant Lainiotis Filter (DTILF)

Central Level

Matrix Operation	Matrix Dimensions	Calculation Burden
$P(k/k)O_n$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$I + P(k/k)O_n$	$I(n \times n) + (n \times n)$	n
$[I + P(k/k)O_n]^{-1}$	$(n \times n)^{-1}$	$(16n^3 - 3n^2 - n)/6$
$F_n[I + P(k/k)O_n]^{-1}$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n[I + P(k/k)O_n]^{-1}P(k/k)$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$F_n[I + P(k/k)O_n]^{-1}P(k/k)F_n^T$	$(n \times n) \cdot (n \times n)$ *symmetric*	$n^3 + 0.5n^2 - 0.5n$
$P(k+1/k+1) = P_n$ $+F_n[I + P(k/k)O_n]^{-1}P(k/k)F_n^T$	$(n \times n) + (n \times n)$ *symmetric*	$0.5n^2 + 0.5n$
$F_n[I + P(k/k)O_n]^{-1}x(k/k)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$\sum_{i=1}^{P} b_i(k+1)$	$(n \times n) \cdot (n \times 1)$	$(P-1)n$
$P(k+1/k+1)\sum_{i=1}^{P} b_i(k+1)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$x(k+1/k+1) = F_n[I + P(k/k)O_n]^{-1}x(k/k)$ $+P(k+1/k+1)\sum_{i=1}^{P} b_i(k+1)$	$(n \times 1) + (n \times 1)$	n
	Total Central Level	$(58n^3 + 9n^2 - 7n)/6$ $+Pn$

Local Level

Matrix Operation	Matrix Dimensions	Calculation Burden
$b_i(k+1) = H_i^T(k+1)R_i^{-1}(k+1)z_i(k+1)$	$(n \times M) \cdot (M \times 1)$	$2nM - n$
	Total Local Level	$2nM - n$

Level	Calculation Burden
Central Level	$(58n^3 + 9n^2 - 7n)/6 + Pn$
Local Level	$2nM - n$
Total	$(58n^3 + 9n^2 - 13n)/6 + Pn + 2nM$

Distributed Steady State Lainiotis Filter (DSSLF)

Central Level

Matrix Operation	Matrix Dimensions	Calculation Burden
$\sum_{i=1}^{P} d_i(k+1)$	$(n \times 1) + (n \times 1)$	$(P-1)n$
$A_{ss}x(k/k)$	$(n \times n) \cdot (n \times 1)$	$2n^2 - n$
$x(k+1/k+1) = A_{ss}x(k/k) + \sum_{i=1}^{P} d_i(k+1)$	$(n \times 1) + (n \times 1)$	n
	Total Central Level	$2n^2 - n + Pn$

Local Level

Matrix Operation	Matrix Dimensions	Calculation Burden
$d_i(k+1) = \overline{P}H_i^T(k+1)R_i^{-1}(k+1)z_i(k+1)$	$(n \times M) \cdot (M \times 1)$	$2nM - n$
	Total Local Level	$2nM - n$

Level	Calculation Burden
Central Level	$2n^2 - n + Pn$
Local Level	$2nM - n$
Total	$2n^2 - 2n + Pn + 2nM$

Received: December, 2008