ICAS Paper 80 - 6.1

A Paper Presented in Session 6.0 of the 12th Congress of the International Council of the Aeronautical Sciences, Munich, October 1980

FFA Library

DESIGN OF NAVIGATION ESTIMATION ALGORITHMS FOR IMPLEMENTATION ON A MICROPROCESSOR [1]

V.B. GYLYS, TEXAS INSTRUMENTS INCORPORATED [2]

J. N. DAMOULAKIS, LEAR-SIEGLER, INC. [2]
(FORMERLY TEXAS INSTRUMENTS INC.)

ABSTRACT

This paper surveys the current techniques used in implementation of estimation algorithms for avionics navigation systems. A general model of guidance, navigation, and control processes is introduced. Recursive estimation of the craft's state is identified as the central and critical function of a multisensor, multiple measurement navigation system. Kalman filters are introduced as the most commonly used estimation algorithms. The implications of choosing a microcomputer as the basic hardware building block on the design of navigation real time software are examined. Partitioning of workload into concurrent and interacting processes is introduced as the basic technique for overcoming the constraints imposed by the hardware. Two levels of partitioning are defined. On the lower level, the conventional form of an estimation algorithm, such as a Kalman filter, may be drastically changed by its decomposition into concurrent processes. Several basic techniques for partitioning the navigation functions in general and the estimation algorithms in particular are illustrated. Other interesting issues and problems -- such as the screening of outlier measurements of the modeling of system statistics are briefly mentioned.

1. INTRODUCTION

In his recent and informative review of new tricks in avionics, Smyth (1980) states that "great changes, propelled by microprocessor technology, are sweeping avionics and control" and that they will continue in the decade ahead." In the introduction to that survey, the author lists the following factors and technologies which, according to him, will most likely influence the next generation of systems: "NAVSTAR/GPS, ring-laser gyros, air-traffic-control update, digital fly-by-wire, full-authority digital propulsion control, flat-panel displays and integrated data-control centers, modular and distributed avionics architecture, cost of airborne data processing, digital buses including potential of fiber optics, modern control theory including Kalman filters and optimal-state estimation, direct digital synthesis, high-level software languages, VLSI/VHSI circuits, and actuation and power-generation devices made possible by new rare-earth magnetic materials." Subsequently, that review proceeds to survey in some detail avionics and controls in the following five broadly applicable technologies:

(i) Flight-path-management technology (navigation, guidance, and air traffic control), (ii) automatic control systems (autopilots, engine controls, and active controls), (iii) crew station technology (displays and controls), (iv) integration and interfacing technology (system architecture and intrasystem communications), and (v) fundamental technologies (analysis and synthesis techniques, software methodology, sensors, etc.).

The present paper is a tutorial survey which addresses only a narrow cross section of the above listed technologies and proceeds to do it at a slow pace. Furthermore, to preserve its own tutorial nature, the paper refers only to readily accessible sources such as introductory texts or journal articles.

As the title of the present paper suggests, it is a survey of techniques used in design of navigation estimation algorithms for implementation on a microprocessor (or more generally, on small computers). Such algorithms are designed as an integral part of the navigation real time software. Thus, the paper also addresses general problems of navigation software design.

Typical microprocessor system architectures under consideration are constructed from small building blocks (microcomputers, standalone, direct access memory units, and digital interface units, the latter needed for communications with the "outside" world) which are interconnected by a single or

---

[1] This paper is based on general experience and observations (as well as on publicly accessible literature) accumulated by the authors while working in the area indicated by its title. The paper conveys the personal views of its authors and does not necessarily reflect the views of the authors' employers or their customers.

[2] Authors' addresses:

V.B. Gylys, M/S 3418, P.O. Box 222013, Dallas, TX 75222,USA. J.N. Damoulakis, Lear-Siegler, Inc., Grand Rapids, MI 49508, USA. propagation

several bus systems. Smyth (1980) mentions the DAIS and the Draper Laboratory Fault-Tolerant Multiprocessor as prototypes of such new architectures. We shall loosely refer to such architectures formed from "small" building blocks as distributed microprocessor systems. Consequently, the paper starts (Section 2) with a brief review of software engineering problems associated with the development of real time software for distributed microprocessor systems.

Since a typical building block of a distributed system, in particular a processor element, is "small", partitioning of navigation functions into concurrently executable and interacting processes is the fundamental technique for fitting software to hardware so as to satisfy the real time processing constraints of the system. This partitioning, as described in the sequel, may have to be performed on two levels: on the higher (natural partitioning) level various functions are grouped into processes without decomposing individual algorithms; on the lower (algorithm decomposition) level, individual algorithms are decomposed. Thus, for example, an estimation algorithm, such as a Kalman filter, is transfigured on the lower level into a set of concurrent algorithms. Such a transfigured form of algorithm may loose all desirable properties of the original algorithm.

To partition the workload properly, one must understand the processing functions to be performed, their real-time constraints, and the interactions among these functions. Hence, the paper proceeds (Section 3) with a brief overview of guidance, navigation, and control processes to provide a perspective for navigation functions. Next recursive estimation of the craft's state, which is usually performed by a Kalman filter, is identified as the critical navigation function on which the stability and the integrity of navigation depend. The criticality of the recursive state estimation function becomes especially evident in multisensor, multitype-measurement navigation systems (Section 6). In such systems, this function synergetically combines diverse types of measurements to optimize (or to improve statistically) the quality of estimates, and at the same time to increase the stability and the robustness of the estimation process. Section 5 summarizes some of the more commonly used techniques for mechanizing recursive estimation schemes and for partitioning the functions of such schemes into concurrent processes. Two examples illustrate cases of lower level partitioning (algorithm decomposition), in which an extended Kalman filter is decomposed into two concurrent processes.

Finally, Section 6 mentions several outstanding problems and issues not directly associated with partitioning, such as prefiltering of measurements in order to screen out the outliers, stabilization of the estimation process, and adaptive estimation of model statistics.

2. IMPLICATIONS OF THE USE OF MICROPROCESSORS ON SOFTWARE DESIGN

The use of a distributed microprocessor system, formed from "small" elements (microcomputers, memory and interface units, all of them interconnected by a bus) has implications not only on the design of real time software but also on the development methodology of such software. Before embarking on our main theme, we briefly examine some of these implications.

The most obvious and at the same time the most far reaching implication is that the workload must be distributed over the microprocessors of the system. Consequently, the computational procedures must be broken up into concurrently executable, interacting processes.

The term "concurrently" in the present context means either "interleaving in time" or "simultaneously" or both. The term "process" throughout this paper (except for Section 3) is used in the same sense as in the operating system theory: It refers to a program in execution or to one which is waiting to be executed, and encompasses the environmental changes caused by execution; it also implies that the program is visible to (i.e., schedulable by) the operating system.

As already stated in Section 1, partitioning may have to be performed on two levels. On the higher level, algorithms and procedures are "naturally" grouped into autonomously schedulable computer programs without decomposing algorithms into smaller, concurrently executable units. On the lower level, such a decomposition of an algorithm (or procedure) is carried out, which has implications on the performance (accuracy, stability, etc.) of the algorithm.

Besides the problems of algorithm performance, the partitioning of workload has other implications as well:

(a) Timing and sizing of software become an important exercise that may have to be reiterated several times during the partitioning process.
(b) An appropriate operating system (or a real time executive) is required to control processes in real time.
(c) Communications among interacting processes must be defined and implemented so as to guarantee the integrity of data accessed by several processes.
(d) Software testing and integration, as well as prediction of performance, have to be performed by simulations, some of which require special equipment.
(e) Presently used higher-order programming languages are not equally suitable for the described structuring of workload.

Principles and Methods of Partitioning

During the past decade there was a great deal of interest in software design for distributed microprocessor systems. For example, Jensen and Boebert (1976) and Gylys and Edwards (1976) describe a methodology and several principles, such as the minimization of the resulting bus traffic, to be used in partitioning. The accumulating experience of the last few years indicates that the most important contributors to good partitioning are (i) the "physical" understanding of the real time system which one is trying to design and (ii) the availability of automated aids to facilitate the

bookkeeping work associated with timing and sizing exercises.

Until the trial versions of basic algorithms, coded in a high order language, become available, the processor and memory loadings can be predicted by analytical models. Biermann(1977) describes such a model for Kalman filters. After the trial versions of software become available, one can harness the compiler to decompose program units into segments, each segment ending with a branch statement, and to estimate the execution times of individual segments. All this can be done as a by-product of compilation.

Real Time Control of Process & Process Communications

The concept of process state control as well as other related concepts introduced by recent advances in operating system theory greatly simplify the designer's task. In a nutshell, intelligent application of these concepts makes it easy for the designer to define at an early time, all possible process states, the rules for transitions among these states, the routines for executing the transitions, the protocol for accessing global data, etc. Furthermore, he (she) can do all these things without binding a priori his design to any rigid process control scheme. This enables one to try out a variety of partitioning schemes throughout the development cycle. Two important issues about which the designer must decide at an early time are (i) the degree of centralization in process control and (ii) the overall synchronization of the distributed system. One recommended and simple approach is to decentralize the process control by providing each microprocessor with an autonomous real time executive and to synchronize the overall execution of process by means of timer interrupts broadcast to the entire system and by means of data interchange.

The following references are current American texts on modern theory of operating systems: Coffman, Jr., and Denning (1973); Coffman, Jr. (1976); Graham (1975); Hansen (1973). Graham gives a very lucid introduction to process states and their control

Hardware

The terms "microprocessor" and "microcomputer" are used loosely in literature. In this paper, the term "microcomputer" emphasizes that the microprocessing element under consideration has its own memory unit. In real time applications (even when several microprocessors are combined as a distri - buted system) this memory typically used to store executable code and perhaps also local variables upon which this code operates.

The present trend in the navigation applications that require a non-trivial amount of processing is toward the use of 16-bit microprocessors. Davis (1979) compares three typical 16-bit microprocessors: Texas Instruments 9900, Zilog 8000, and Intel 8086.

Another issue is the availability of hardware implemented floating point arithmetic. It is very time consuming to develop and subsequently modify navigation software if fixed point arithmetic is used. Therefore, the current trend is toward hardware implemented floating point arithmetic, perhaps in the form of a "special box." If this cannot be done then firmware (as a second choice) or interpreted (as a third choice) floating point arithmetic is used.

3. FUNCTIONS OF A NAVIGATION SYSTEM

Guidance, Navigation, and Control

We start with a brief review of the basic problems of navigation. Navigation starts with definition of a destination and selection of a route. A variety of different factors -- such as minimization of fuel consumption or travel time, avoidance of collisions, passing through (or over) preselected way points -- may influence the selection of a route. Furthermore, one may want to modify the selected route in real time. After the craft starts moving toward the specified destination, its progress (route and velocity) must be maintained throughout the journey. The maintenance of the progress is a continuous, repetitively executed process. Each repetition of this process typically consists of four steps or functions, executed in the indicated order: (1) determination of the craft's position, velocity, and perhaps acceleration at the time of such a repetition; (2) prediction of the craft's future progress based on its present state (i.e., position, velocity, and acceleration); (3) computation of the corrective action (i.e., acceleration) required to maintain the selected route or at least to bring the craft to the specified destination; (4) execution of the computed corrective action by appropriate handling of the craft.

The functions associated with selection and real time modifications of the route are known as the guidance functions. The functions constituting steps 1 and 2 of an instance of the above described, progress maintainance process deal with determination of the craft's progress and are referred to as the navigation functions. Finally, the functions constituting steps 3 and 4 are concerned with handling of the craft and will be called the control functions.

These processes of guidance, navigation, and control lead to a system of three interacting feedback loops, with the craft (or more precisely, with the progress of the craft) linking all these loops together. Pictorially these ideas are summarized in Figure 3-1 which contains canonical model of guidance, navigation, and control processes in a craft.

It is worthwhile to note that many existing guidance and control systems, especially those in smaller missiles, are a special case of the canonical model shown in Figure 3-1. In such systems, guidance and control function without explicit knowledge of position; i.e., only two out of three feedback loops - the guidance loop and the craft control

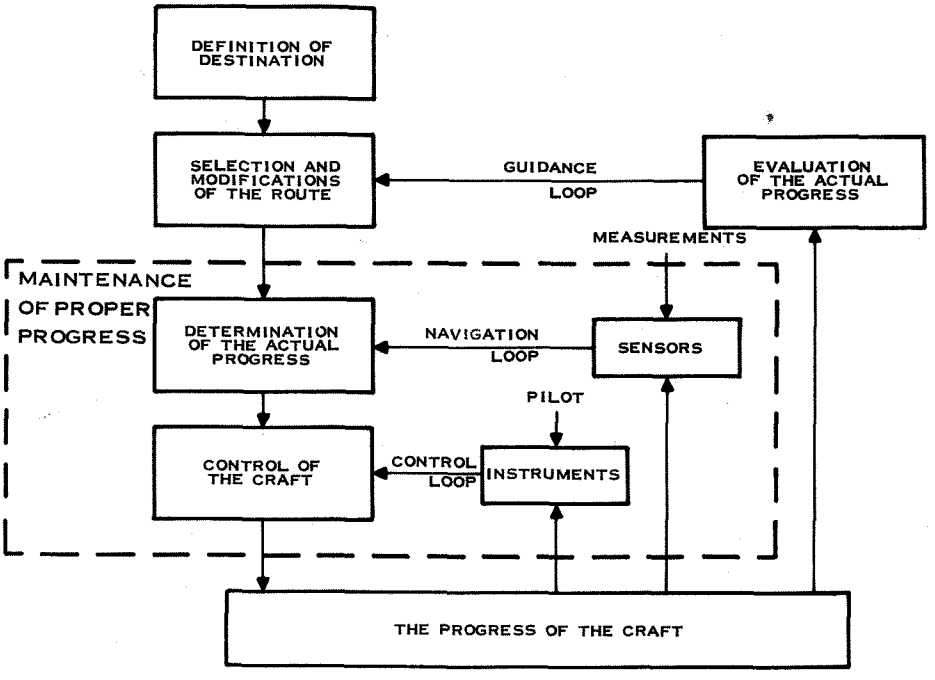

Figure 3-1. A Canonical Model of Guidance, Navigation and Control Processes in a Craft.

loop -- are present in such systems. An extreme case of the canonical model is a stationary surveying system which is concerned solely with determination of position. Such a system operates only with a single loop - the navigation loop.

Clearly, Figure 3-1 abstracts an actually more complex situation. In many actual systems, the boundaries dividing the three processes (guidance, navigation, and control) are fuzzy; also, the interactions among these processes may be more complex and cannot be represented by means of three nested feedback loops as in Figure 3-1.

A similar and succinct view of navigation is given by E.W. Anderson (1966). Basic problems of navigation are also discussed in the first two chapters of Kayton and Fried (1969). Farvell (1976) is a recent reference on integrated aircraft navigation.

Navigation Functions

From the viewpoint of our primary interest in the data processing aspects of navigation functions, it is convenient to divide them into two major functions: (1) recursive estimation of the (craft's) state and (2) relative navigation. The latter function operates on the information provided by the estimation function and includes tasks such as prediction of the craft's future course, computation of waypoints, and various coordinate transformations. Relative navigation is also the function which transforms the information contained in estimates of the craft's state to the outputs which are to be displayed to the pilot. The algorithms used in relative navigation computations are well known: E.W. Anderson (1966) and Kayton and Fried (1969) are handy references that contain further references to the available literature.

The computations performed by relative navigation may be viewed as being a by-product of recursive state estimation. Thus, the integrity and the stability of navigation process mainly depend on the quality of state estimation. With good estimates of the Craft's state, design of the relative navigation alogorithms for implementation as real time software becomes a problem of numerical analysis, computer memory, and processing rates. Inadequately low processing rates (due to insufficient processor resources) may produce aged or inaccurate navigation outputs, but they will not blow up the basic navigation process. However, in certain applications of navigation outputs, such as the weapon delivery computations, insufficient processing rates may cause serious problems.

The basic model of the navigation system which we shall use consists of a recursive state estimator, a relative navigation subsystem, a control subsystem, and interfaces. This model is pictorially summarized in Figure 3-2. We write s to denote the state vector and m - the external measurements that help estimate the state.

The only type of estimation schemes considered are the so-called discrete state, discrete (sampled) measurement schemes. (Thus, the value of a vector or scalar x at time t_k will be denoted by x(k).) In navigation work, the components of state vector s usually model at least the position coordinates and the velocity components of the craft. In addition, extra components may be used to model the craft's acceleration and to model the uncer-

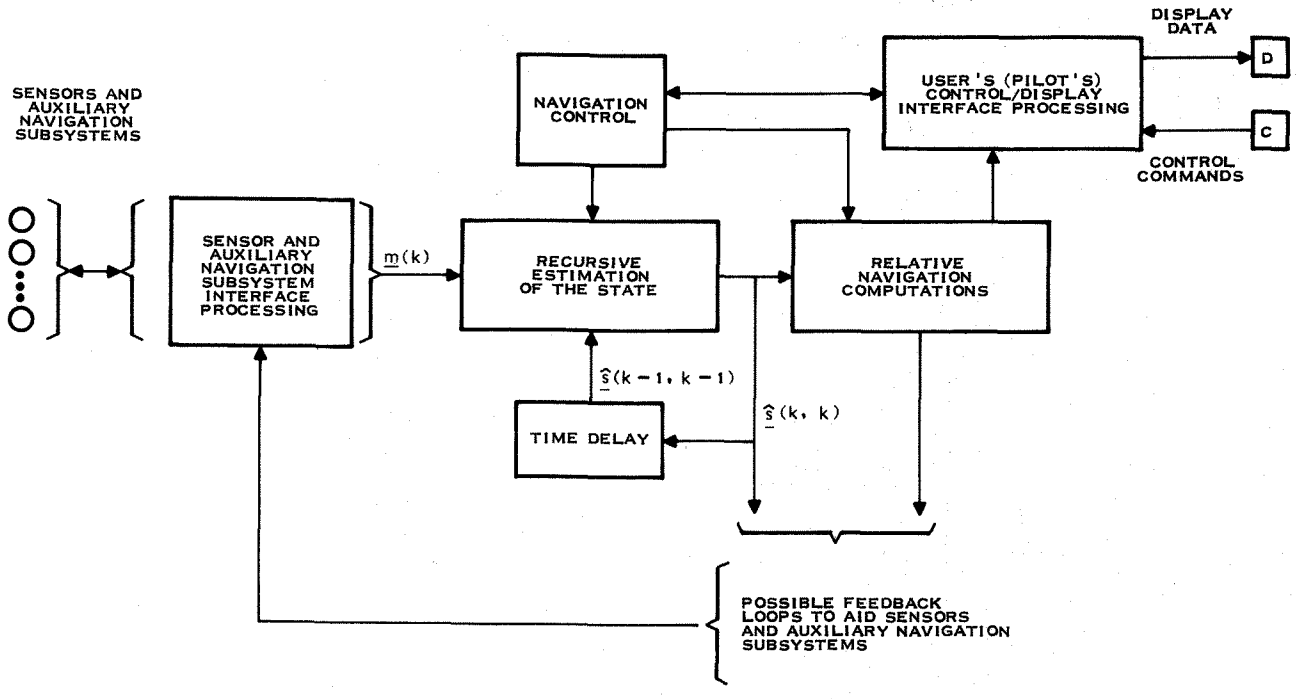

Figure 3-2. Major Data Processing Functions of and Data Flow Through an Integrated Navigation System Operating on Multisensor and Multitype Measurements.

tainties and errors in various components of the navigation system itself or in sensors and systems that furnish external measurements to the navigation system.

As an example, consider the situation where the inertial system of the craft furnishes the craft's velocity, acceleration, and attitude measurements to the recursive state estimation process, and where in order to recalibrate the inertial system, this process estimates the platform misalignment and gyro drift rate. Maybeck (1979) in Chapter 6 discusses on a tutorial level this example in detail.

4. INTEGRATED NAVIGATION SYSTEMS

The perception of a navigation system changed drastically during the last decade. We are moving away from single sensor type, limited coverage, and limited accuracy navigation systems to integrated navigation systems. These new systems operate by combining synergetically the measurements coming from several different types of sources (obtained through use of various sensors) or from other navigation systems (such as inertial systems) to produce navigation estimates of superior quality and integrity. In addition, some of these new systems are designed to work without any geographic restrictions, without deterioration in time, and under the most difficult environment conditions. These new types of navigation systems were made possible through the advances of the last decade in digital technology and software methodology, through the emergence of new types of navigation data generators (such as the NAVSTAR Global Positioning System) and through the accumulation of knowledge and experience in the design of recursive estimators (such as Kalman filters) for real time systems.

What characterizes a modern integrated navigation system is that it is built around a recursive state estimator (which usually contains a Kalman filter) operating in time domain. We refer to such a system as the multisensor, multitype-measurement (or just the integrated) navigation system. The block diagram contained in Figure 3-2 summarizes the structure of such an integrated navigation system.

James L. Farrell (1976) is a recent text on integrated aircraft navigation, written from the systems engineering viewpoint.

5. MECHANIZATION OF RECURSIVE STATE ESTIMATORS

Kalman Filters

Several standard schemes for mechanizing the recursive estimator in navigation systems and for partitioning the resulting workload are introduced in the present section. Since such a recursive estimator is usually built around some form of a Kalman filter, the basic equations of a Kalman filter (including the modeling assumptions) are reviewed first. This is done only for the case of a discrete (state) linear system with sampled measurements. These equations and assumptions are summarized in Table 5-1. Modeling assumptions and estimator algorithms for more complex or general

TABLE 5-1

KALMAN FILTER ALGORITHM FOR A
DISCRETE LINEAR SYSTEM
WITH SAMPLED MEASUREMENTS

A. SYSTEM MODEL

1. Propagation of the system state vector from $t=t_{k-1}$ to $t=t_k$:

$$s(k) = F(k, k-1)s(k-s) + G(k-1)w(k-1)$$

2. Measurements at $t=t_k$:

$$m(k) = H(k)s(k) + v(k)$$

3. Initial Conditions at $t=t_0$:

$$E[s(0)] = \hat{s}(0), \quad Cov[s(0) - \hat{s}(0)] = P(0)$$

4. Assumptions about system statistics:
The processes $\{w(k)\}$ and $\{v(k)\}$ are zero mean, independent Gaussian processes with covariances

$$E[w(k)w(j)^T] = Q(k)\,\delta_{kj}$$

and

$$E[v(k)v(j)^T] = R(k)\,\delta_{kj}$$

Furthermore,
$s(0)$ is independent of $w(k)$ and $v(k)$ for any k.

B. ESTIMATION PROCEDURE

5. Propagation of estimates from $t=t_{k-1}$ to $t=t_k$:

$$\hat{s}(k)^- = F(k,k-1)\hat{s}(k-1)^+$$

$$P(k)^- = F(k,k-1)P(k-1)^+F(k,k-1)^T + G(k-1)Q(k-1)G(k-1)^T$$

6. Updating of estimates at $t = t_k$:

$$K(k) = P(k)^-H(k)^T\,[H(k)P(k)^-H(k)^T + R(k)]^{-1}$$

$$P(k)^+ = [I-K(k)H(k)]\,P(k)^-$$

$$\hat{s}(k)^+ = \hat{s}(k)^- + K(k)\,[m(k) - H(k)\hat{s}(k)^-]$$

Notation used above:

(1) Upper case letter represent matrices.

(2) Lower case letters represent scalars or column vectors.

(3) "x(k)" represents the value of column vector (or scalar) x at $t = t_k$.

(4) "A'" represents the transpose of matrix A; if x is a column (row) vector, then "x'" represents the transpose of x, which is a row (column) vector.

---

cases (for example, for a system in which the propagation of state or the state-to-measurement transformation is nonlinear) are discussed in references such as Anderson and Moore (1979) and Schmidt (1976).

Direct and Indirect Formulations of the Kalman Filter

Two different alternative approaches are used for formulating the Kalman filter in a navigation system. With the first approach, which is called direct (or total state) formulation, the state vector s of the Kalman filter represents the craft's (navigation) total state. Thus, the Kalman filter directly estimates the total state vector s. In this case, s has three components which represent the craft's position coordinates, three components which represent the craft's velocity, etc.

With the second approach, which is called indirect (or state error) formulation, the craft's (navigation) total state vector is not directly estimated by the Kalman filter. Instead, the Kalman filter now estimates the state error ds. This estimate of ds is then subtracted from the best available value of s in order to obtain an estimate of s.

The indirect formulation of the Kalman filter is very convenient in multisensor, multitype-measurement navigation systems. If one has two streams of measurements coming in at two very different rates, then the fast rate stream may be used for rapid propagation of the craft's navigation state s (there may be good reasons for requiring a high propagation rate), whereas the slow rate measurement stream for driving the Kalman filter. Here, we tacitly assume the suitability of each measurement streams to perform the functions assigned to it.

Figure 5-1 illustrates this type of mechanization for the case where the fast rate measurements are the velocity and acceleration outputs of an inertial navigation system and the low rate measurements are GPS space vehicle range and range rate measurements. This example also illustrates another point: it suggests a natural partitioning of the recursive state estimator of the considered navigation system into two concurrent processes. The total navigation state is propagated by the high rate process; the state error is estimated by the Kalman filter, which would be implemented as the low rate process.

Maybeck (1979) in Chapter 6 compares the direct and indirect formulation of a Kalman filter for navigation applications. Information on GPS can be found in Henderson (1980), Milliken and Zoller (1978), Van Dierendonck et al.(1978). Cox, Jr. (1978) discusses integration of GPS with inertial navigation systems.

Open Loop and Closed Loop Mechanizations

In addition to the two formulations of the Kalman filter according to the type of the state vector that is actually estimated by the filter (discussed in the preceding subsection), mechanization of the navigation recursive estimator may also differ depending on whether the estimation outputs are fed back or not to aid some of the sensors or external subsystems that are generating measurements for the navigation system. Thus, one can speak of open loop (also called feedforward) mechanizations and closed loop (or feedback) mechanizations. Each of these two mechanizations may be combined with any of the two formulations (direct or indirect) of the Kalman filter.

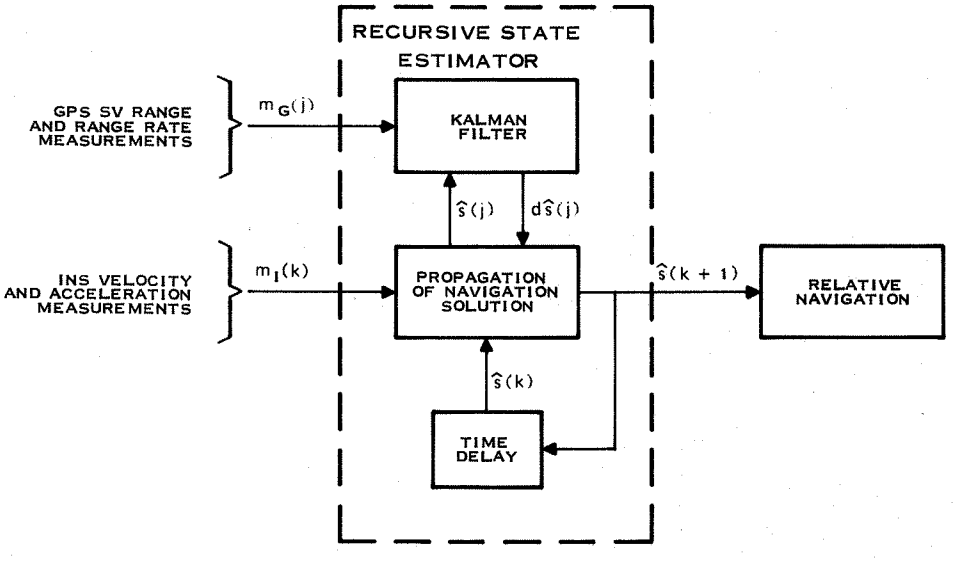

Figure 5-1. Indirect Formulation of a Kalman Filter for a Recursive Estimator Partitioned into Two Concurrent Processes. Note that $d\hat{s}$ must be propagated (perhaps interpolated) and subtracted from $\hat{s}$ by the fast rate propagation process. This is an example of natural partitioning if the slow rate Kalman filter process is not further partitioned.

As noted earlier, Maybeck (1979) in Chapter 6 gives gives example of indirect feedforward and indirect feedback mechanizations in which the fast rate measurements come from an inertial navigation system. In his indirect feedback mechanization, the estimates produced by the Kalman filter are used to correct the platform misalignments and gyro drift rates of the inertial navigation system. The author also compares the cons and pros of each of these two types of mechanization.

Another type of feedback mechanization, discussed by Gylys and Ward (1980), is the case where the navigation estimates are used to aid GPS receivers. That example is also interesting from the viewpoint of Kalman filter formulation (a combination of direct and indirect formulations is used) and workload partitioning (which will be briefly mentioned in the sequel).

Piecewise Recursive Mechanizations

Even a cursory examination of linear Kalman filter equations immediately reveals to anyone familiar with matrix operations that the costliest portions of the Kalman filter algorithm (with regard to the processor time and memory requirements) are propagation and updating of the state error covariance matrix P and computation of Kalman gains K. If the navigation system under design is supposed to operate only on measurements from sources of a single type, such as the GPS range and range rate measurements, and if, in addition, these measurements must be processed at a faster rate than the available processor resources allow, then the so-called piecewise recursive mechanization of the filter offers one (although perhaps not a perfect) solution to the designer's problem. This mechanization also offers an interesting partitioning of the workload.

Piecewise recursive mechanization works as follows. The Kalman filter algorithm is decomposed into two concurrent processes. (Note that according to the terminology introduced earlier, this constitutes a lower level of partitioning, which may destroy some of the desired characteristics of the algorithm expressed in its original form). During each execution cycle, one of these processes (that one which is executed at a fast rate) propagates the estimate of the state vector s, processes the measurements, computes the measurement residuals, scales these residuals by multiplying them with Kalman gains K (the latter computed at a slower rate by the other process), and updates the estimates of s. The other process runs at a considerably slower rate than the first one: each cycle of the slowly running process spans, say, 10 to 20 cycles of the fast running process. In each execution cycle, the slowly running process first propagates the state error covariance matrix P, then computes the Kalman gains K and updates P while utilizing the information sent to it by the fast running process. This scheme can be implemented on a single microprocessor by making the fast running process a foreground process which is scheduled on strictly periodic basis, while letting the slowly running process a background process which gets all processor time that remains after the execution of the foreground process, executive functions, and ad hoc scheduled special tasks. Naturally, the microprocessor under consideration for this scheme must have enough throughput, which often means that the foreground process will require less than 40 to 50% of the total available time.

It appears that piecewise recursive estimation was first proposed by Dressler and Ross (1979). Applications of this scheme to actual navigation problems are described by Gylys and Ward (1980) and also by Upadhyay and Damoulakis (1980).

6. OTHER OUTSTANDING DESIGN ISSUES

There are many other issues besides partitioning which navigation system designers face. Some of these issues, which by no means are characteristic only to microprocessor implementation, are briefly examined in the present section.

Adaptive Estimation

Adaptive estimation is concerned with estimation of the unknown parameters of the system model (upon which the estimator is built) in addition to the estimation of system states. If a parameter of the system dynamics model is unknown, such as an aerodynamic drag coefficient, it can be estimated by modeling it in the state vector. This increases the processing resource requirement. In many situations, an a priori model of such a parameter, perhaps expressed as a function of other known parameters or state variables and obtained by means of simulation, is sufficient.

One situation which quite often occurs is where some or all of the covariances of the process or measurement noise are unknown and cannot be easily modeled. This problem has interested many researchers since the inception of Kalman filters, but no satisfactory solution for all cases exists. Typical adaptive techniques for this problem are based on observation and analysis of residual sequences. Under favorable conditions, the residuals (or their squares) form a stochastic process with known distributional properties. Thus, at least in theory, one can perform a statistical hypothesis test to decide whether the behavior of residuals is reasonable. If the hypothesis that the residuals are reasonable is rejected, then the decision making process faces a dilemna when nothing is known about both types of covariances (i.e., the process noise and measurement noise covariances), because in such a situation covariances of both types or only of a single type may be incorrect. Even if this approach to modeling the noise statistics were reliable, it could not be used in situations where the noise characteristics are changing faster than they can be adaptively estimated with the available processing resources.

A reasonable alternative to determination of noise statistics covariances is to require that the sensors and the other systems which send their measurements to the navigation system also include the measurement quality assurance code together with the measurements. Signal-to-noise ratio is an example of such a quality assurance code. This type of information is useful not only for determi-

nation of the noise statistics model, but also for detection of the leading and trailing edges of bad data bursts.

Recent references on the problems of adaptive estimation and system model identification are Brewer (1976), Leondes and Pearson (1973), and Ohap and Stubberud (1976).

Preserving the Stability of Navigation Process

In many navigation systems, the stability of the navigation process and the integrity of the outputs produced by the process are critically important to the pilot of the craft or to the automatic control systems of the craft. If the navigation system operates on measurements only from sources of a single type and if, for example, the sensor equipment starts to malfunction intermittently, then such malfunctioning episodes will manifest themselves as bursts of completely bad data. It is then extremely important for the recursive estimator of the navigation system to detect the leading and trailing edges of such bursts in order to stop incorporation of measurements into the navigation solution as soon as they go bad and to stop discarding them as soon as they become acceptable. One approach is to have each measurement accompanied by the quality control code (to be generated by the sensor) which will help the recursive estimator decide, perhaps with some assistance from the time series generated by the measurement residuals, whether a measurement is acceptable or not.

It is easier to tackle this problem in the case of multisensor, multitype-measurement navigation. Then in such a situation, measurements of one type (or source) can be used to test the measurements of the other type (or source).

If the recursive navigation solution starts to destabilize itself due to a short-lived, intermittent burst of bad data, it is still sometimes possible to save the solution by censoring the estimates. For example, if the craft is a missile and if one a priori knows approximately what accelerations it will undergo during the flight, one can replace the estimates of acceleration with the a priori established bounds each time when these bounds are exceeded. This principle, known to statisticians, is used in the so-called robust estimation procedures. Naturally, censoring may also destroy any remaining optimality.

Numerical Roundoff Errors and Filter Instability.

The processing and storage constraints of microprocessor environment usually force the use of minimal precision digital arithmetic, such as "single precision" floating point arithmetic. Certain parts of the Kalman filter algorithm, especially the state error covariance update equation (the second equation in Part 6 of Table 5-1), are vulnerable to the destabilizing effects of roundoff errors.

Fortunately, this numerical roundoff problem has been widely researched and reported in literature. Stable and efficient algorithms for the updating of state error covariances P and computation of Kalman gains K have been constructed. Loosely, these algorithms are known as "square root filtering," a term originating from J.E. Potter who was one of the first contributors to the solution of this problem. Recent variants of the square root filtering algorithms do not require computation of square roots. Bierman (1977) is a good, up-to-date reference for further information on these algorithms.

7. CONCLUSIONS

This paper surveyed, on a very elementary and tutorial level, characteristic problems faced by a person who tries to design (mechanize) navigation estimation algorithms for implementation on a distributed microprocessor system. These problems were examined mainly from the viewpoint of the designer of real time software. The need to partition the algorithms into concurrently extendable, interacting processes was identified as one of the main implications of the use of microprocessors. This partitioning may have to be carried out on two levels: on the lower level, the original forms of algorithms may become destroyed. It was noted that efficient software timing and sizing tools and good understanding of the "physics" of the navigation problem at hand are essential to succeed in this work.

ACKNOWLEDGMENTS

The authors wish to express their gratitude to the staff of Texas Instruments Incorporated for the support received during the preparation of this paper. We are particularly indebted to Srini Raghavan for advise and assistance, Howard Glewwe for editing and expediting the paper almost in zero time, and to Ruth Wright, Marie McManus, and Shirley James for their typing expertise. All shortcomings and errors point back to the authors.

Finally, the first author expresses his gratitude to Gary Boswell and Weldon Word, both at Texas Instruments Incorporated, for allowing him to present this paper to the 12th Congress of ICAS.

REFERENCES

Anderson, B.D.O, and Moore, J.B. (1979). Optimal Filtering. Prentice-Hall, Inc., Englewood Cliffs, N.J.

Anderson, E.W. (1966). The Principles of Navigation. Hollis and Carter, London.

Bierman, G.J. (1977). Factorization Methods for Discrete Sequential Estimation. Academic Press, New York.

Brewer, H.W. (1976). "Identification of the Noise Characteristics in a Kalman Filter." Control and Dynamic Systems, Vol. 12 (edited by C.T. Leondes). Academic Press, New York.

Coffman, Jr., E.G., and Denning, P.J.(1973). Operating Systems Theory. Prentice-Hall, Inc., Englewood Cliffs, N.J.

Coffman, Jr., E.G. (ed.) (1976). Computer and Job-Shop Scheduling Theory. John Wiley and Sons, Inc., New York.

Cox, Jr., D.B. (1978). "Integration of GPS with Inertial Navigation Systems." Navigation, Vol. 25, No. 2, pp. 236-245.

Davis, H.A. (1979). "Comparing Architectures of Three 16-Bit Microprocessors." Computer Design, July '79, pp. 91-100.

Dressler, R.M. and Ross, D.W. (1970). "A Simplified Algorithm for Suboptimal Nonlinear State Estimation." Automatica, Vol. 6, pp. 477-480.

Farrell, J.L. (1976). Integrated Aircraft Navigation. Academic Press, New York.

Graham, R.M. (1975). Principles of Systems Programming. John Wiley and Sons, Inc., New York.

Gylys, V.B., and Edwards, J.A. (1976). "Optimal Partitioning of Workload for Distributed Systems." Digest of Papers, COMPCON 76, IEEE Computer Society, pp. 353-356.

Gylys, V.B., and Ward, P.W. (1980). "Design and Performance of the Missile-Borne Receiver Set." A paper presented at NAECON 1979; reprinted in Equipment Group Journal, Vol. 3, No. 3, Texas Instruments Incorporated, Dallas, Texas.

Hansen, P.B. (1973). Operating System Principles. Prentice-Hall, Inc., Englewood Cliffs, N.J.

Henderson, D.W., and Coriat, H. (1980). "Status Report - Global Positioning System." Navigation, Vol. 27, No. 1, pp. 54-64.

Jensen, E.D., and Boebert, W.E. (1976). "Partitioning and Assignment of Distributed Processing Software." Digest of Papers, COMPCON 76, IEEE Computer Society, pp. 348-352.

Kayton, M., and Fried, W.R. (eds) (1969). Avionics Navigation Systems. John Wiley and Sons, Inc., New York.

Leondes, C.T., and Pearson, J.O. (1973). "Kalman Filtering of Systems with Parmeter Uncertainities - A Survey." International Journal of Control, Vol. 17, pp. 785-792.

Madnick, S.E., and Donovan, J.J. (1974). Operating Systems. McGraw-Hill Book Company, New York.

Maybeck, P.S. (1979). Stochastic Models, Estimation, and Control. Vol. 1. Academic Press, New York.

Milliken, R.J. and Zoller, C.J. (1978). "Principle of Operation of NAVSTAR and System Characteristics." Navigation, Vol. 25, No. 2, pp. 95-166.

Ohap, R.F., and Stubberud, A.R. (1976). "Adaptive Minimum Variance Estimation in Discrete Time Linear Systems." Control and Dynamic Systems, Vol. 12 (edited by C.T. Leohdes). Academic Press, New York.

Schmidt, G.T. (1976). "Linear and Nonlinear Filtering Techniques." Control and Dynamic Systems, Vol. 12 (edited by C.T. Leondes). Academic Press, New York.

Smyth, R.K. (1980). "Avionics and Controls in Review." Astronautics and Aeronautics, Vol. 18, No. 4, pp. 40-52.

Upadhyay, T.N., and Damoulakis, J.N. (1980). "Sequential Piecewise Recursive Filter for GPS Low-Dynamics Navigtion." IEEE Transactions on Aerospace and Electronic Systems, Vol. AE5-16, No. 4, pp. 481-491.

Van Dierendonck, A.J., et al. (1978). "The GPS Navigation Message." Navigation, Vol. 25, No. 2, pp. 95-106.

V.B. GYLYS &
J.N. DAMOULAKIS

ERRATA (1/2)

P. 1 (bottom line of the left-hand column) - delete the last word, "propagation"

P. 2 (2nd paragraph in the left-hand col.) - in the 2nd sentence insert a comma after "... on the higher (natural partitioning) level"

P. 3 (3rd paragraph in the left-hand col.) - the 1st sentence should read:

> The concept of process state control, as well as other related concepts introduced by recent advances in operating system theory, greatly simplify the designer's task.

P. 3 (5th paragraph in the left-hand col.) - the last sentence should read:

> In real time applications (even when several microprocessors are combined as a distri - buted system), this memory is typically used to store executable code and perhaps also local variables upon which this code operates.

P. 4 (3rd paragraph in the left-hand col.) - the 1st word in the last sentence of that par. should be changed from "Farvell" to "Farrell"

P. 4 (2nd paragraph in the right-hand col.) - the 6th word in the 3rd sentence should read "craft's" instead of "Craft's"

P. 5 (3rd paragraph in the right-hand col.) - delete "James L." from the beginning of this paragraph

P. 6 (Table 5-1)

Modify Part 1 of the system model to be:

1. Propagation of the system state vector from $t=t_{k-1}$ to $t=t_k$:
   $s(k) = F(k, k-1)s(k-1) + G(k-1)w(k-1)$

Correct Remark (4) at the bottom of the table to be:

(4) "A^T" represents the transpose of matrix A; if x is a column (row) vector, then "x^T" represents the transpose of x, which is a row (column) vector.

V.B. GYLYS &
J. N. DAMOULAKIS

ERRATA (2/2)

p.7 (4th paragraph in the left-hand column) - the word "streams" in the last sentence should be changed to "stream"

p.8 (1st paragraph in the left-hand col.) - delete the word "gives" on the first line.

p.8 (4th paragraph in the left-hand col.) - the (beginning of the / bottom) sentence should be changed to:

This scheme can be implemented on a single microprocessor by making the fast running process a foreground process which is scheduled on strictly periodic basis, while letting the slowly running process a background process which gets all processor time that remains after the execution of the foreground process, executive

p.8 (4th paragraph in the right-hand col.) - the last word of the 3rd sentence should be changed from "requirement" to "requirements"