On-line state estimation with a small computer

G. C. Coggan and J. A. Wilson

Department of Chemical Engineering, University of Nottingham

This paper shows that quite small computers enable the powerful Kalman sequential estimator, which involves matrix operations, to be applied to industrial processes for state and parameter estimation and hence improved control.

(Received December 1969)

An on-line application of digital computers which is growing rapidly in importance is the supervision or direct control of industrial processes. On economic grounds the mere replacement of conventional analogue controllers by a central processor is rarely justified but when the extra capabilities of a computer are exploited the financial benefits can be considerable.

Optimal control by computer is a major possibility but a serious problem in using plant data for this purpose arises from the sheer inadequacy and inaccuracy of such data. Relatively few process variables are amenable to continuous or frequent measurement and in particular the most important one, product quality, is usually quite inaccessible for control purposes. Mathematical models may be used to predict the values of such variables but, apart from the effects of using imprecise models, the unmeasured random disturbances, which are always evident in industrial processes, render these predictions inaccurate. Furthermore the use of mathematical models by themselves implies that the state of the process was known accurately at some point in time and this is never true.

In a situation where little information is reliable, optimal control may still be feasible provided that 'best' estimates of the current state of the system are available. Thus the problem centres around on-line state estimation. Optimal, sequential, discrete-time estimation techniques due to Kalman (1960) have received much attention recently but the emphasis has often been upon aerospace applications and guidance, e.g. Bucy and Joseph (1968). Coggan and Noton (1969) demonstrated that these techniques could be extended and applied to industrial processes to produce remarkably good estimates of unmeasured variables in adverse circumstances. In some cases the estimates of unmeasured variables were more accurate than the measurements of the measured variables. However, much of their work was done with simulated plants on an IBM 360/75 computer—a machine which is an order of magnitude bigger than anything likely to be used on an industrial process—and one of the principal application problems envisaged by them, and emphasised during subsequent discussions with industrially-oriented control engineers, concerned the hardware requirements for state estimation.

Prior to this investigation it has been commonly assumed that the matrix operations involved in the extended Kalman estimator are simply impractical on the kind of computer used to supervise industrial processes. It is the purpose of this paper to demonstrate that on-line sequential state estimation using a Kalman filter is quite feasible on one of the smallest digital computers in existence.

Notation

$\mathbf{A}'$	The transpose of matrix A
$\mathbf{A} \rightarrow \mathbf{C}$	Matrix A is copied into matrix C
$\mathbf{A}$	Transition matrix
b	Forcing vector
$\mathbf{CQC}'$	Matrix of covariance estimates for random process disturbances
$\mathbf{F}$	Filter matrix
$\mathbf{G}$	Error covariance matrix for the estimates of state variables
$\mathbf{H}$	Measurement matrix
$\mathbf{I}$	The unit matrix
M	The 'order' of the state vector, i.e. the number of state variables
$\mathbf{P}$	Error covariance matrix for the predicted values of state variables
$\mathbf{R}$	Matrix of measurement error covariances
T	Computation time for one pass
x_a	Actual state vector
x_f	'Filtered' or 'estimated' state vector
x_p	Predicted state vector using the process model
y	Vector of measurements
$x_a(k)$	Vector x_a at the kth sample instant
$\bar{x}$	Mean or normal state vector
σ_f^2	Variance of the error of estimation of x_f [obtained from diag (**G**)]
$d(k)$	Estimation error at the kth sample instant.

The computer

The machine in use is a basic PDP 8 having 4K core store of 12-bit words. Backing store is provided by a 184K magnetic tape. The core is split into 32 blocks or 'pages' of 128 words each. Except for page zero indirect addressing must be used between pages. Memory cycle time is 1·5 μs and add time 3·0 μs.

A floating point software package, occupying almost one half of the core store, is used for mathematical operations and floating point input and output. Accuracy is to seven decimal figures.

Matrix pack

This is a set of matrix routines necessary to the operation of the estimator algorithm. Matrices are stored row by row

Table 1 Routines contained in the matrix pack

TITLE	PARAMETERS	OPERATION	STORAGE (WORDS)
RDMAT	**A**	input **A** via the teletype	15
PRMAT	**A**	output **A** via the teletype	26
MSORA	**A, B, C**	**A+B→C** or **A−B→C**	36
MULMAT	**A, B, C**	**AB→C**	39
MATVEC	y, **A**, x	**A**x→y	37
DUPMAT	**A, B**	**A→B**	16
VECMAT	**A**, D	**A**D→**A** (D is a diagonal matrix)	27
TRAMAT	**A, B**	**A**′→**B**	29
INVRS	**A**	**A**$^{-1}$→**A**	120
PLUS I	**A**	**A+I→A**	19

in the core store, i.e. a_{11}, a_{12}, . . . , a_{1n}, a_{21}, a_{22}, . . . etc. Thus a row of matrix elements is stored as a vector. The complete matrix pack, listed in **Table 1**, occupies three pages of store, besides entry addresses and small initialising routines on page zero. All the routines operate in a conventional manner apart from MULMAT and INVRS. MULMAT: For economy in terms of core store the matrix multiplication **C** = **A.B** is carried out by treating **B** as a set of column vectors and using MATVEC to produce corresponding columns of **C**. To do this the columns of **B** must be stored as vectors, i.e. b_{11}. b_{21}. . . . , b_{n1}, b_{12}, b_{22}, . . . etc. Consequently, we must begin by transposing **B** and use MATVEC to produce the column vectors of **C**, which are stored as rows and then transpose the resulting matrix. INVRS: Since space is so critical a pivotal condensation method is attractive, as this generally entails use of little space beyond that occupied by the matrix itself.

For the most accurate results the next pivot chosen should be the largest element which does not lie in the same row or column as a previously chosen pivot. However, this normally requires a rearrangement of rows and columns at the end of the procedure. If instead we pivot about the diagonal elements in sequence no rearrangement is necessary. Thus the storage required for both program and numerical manipulations is reduced significantly. This saving must be weighed against the potential loss of accuracy or even complete failure, which can occur if a diagonal element approaches zero. The probability of this happening seems small in this application.

Storage requirements

The matrix pack (284 words), floating point package (1,553 words) and the estimator program (256 words) occupy a total of 2,193 words of core store. Neglecting page zero and the final page, which contains library loader and binary loader programs, we have 1,647 words available as data space.

Without backup store we require space for five matrices. A floating point number occupies three words, thus limiting the number of state variables (M) to 10, i.e. entier [$\sqrt{(1647/15)}$]. However, to accommodate the required seven vectors in addition to the matrices we must limit M to nine leaving 243 words unused.

Using magnetic tape backup the control software occupies 128 words, leaving 1,519 words as data space. Only three matrices are required simultaneously in the core store and so the largest M we can handle is 12, i.e. entier [$\sqrt{(1519/9)}$].

Any increase in M above 12 would entail the alteration of the matrix addition and multiplication routines so as to operate with sections of the matrices in the core, rather than with complete matrices as at present.

Data transfer

Assuming that a maximum of 12 state variables is to be considered, the most convenient method for transfer of a matrix to or from tape is to transfer four pages of store together, as four is the nearest integral number of pages occupied by a 12 × 12 matrix.

Estimator algorithm

We assume that the actual state of the process to be monitored is given in discrete time by

$$x_a(k+1) = \mathbf{A}.x_a(k) + \text{unmeasured disturbances},$$

whilst the measurements are related to the state by

$$y(k+1) = \mathbf{H}.x_a(k+1) + \text{random errors}.$$

The problem is to produce 'best' estimates $x_f(k+1)$ of $x_a(k+1)$ given $y(k+1)$ and knowing $x_f(k)$.

A more detailed description of the operation of the recursive Kalman estimator, as applied to chemical engineering systems, has been given by Coggan and Noton (1969). Briefly, the algorithm used here can be written:

1:
$x_p = \mathbf{A}x_f + b$
$\mathbf{P} = \mathbf{AGA}' + \mathbf{CQC}'$
$\mathbf{G} = (\mathbf{I} + \mathbf{PH}'\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{P}$
$\mathbf{F} = \mathbf{GH}'\mathbf{R}^{-1}$
$k = k + 1$ (discrete time)
read (y)
read (b)
$x_f = x_p + \mathbf{F}(y - \mathbf{H}x_p)$
write (x_f) best estimate at time k
GOTO 1

Often **H** and **R** are diagonal matrices and so, for economy of space and computer time, have been treated as such (otherwise a large amount of time can be wasted on null arithmetic). If the product **CQC**′ is not time-variant, it may be calculated outside the iterant loop.

Experimental runs

Computation times were recorded for different values of M, by reading in the vector y and then allowing the program to operate for a specified number of iterations, using the same y, before outputting x_f. The number of iterations specified was usually adjusted to give a recorded time on a stopwatch of about one minute (<1% accuracy). The computation time per iteration was apparently independent of both the number of iterations using a single y vector and the numerical magnitude of the elements of the y vector. The results are shown in **Fig. 1.**

The processes modelled and used for these experiments were in fact isothermal gas absorption columns, the state vector being made up of the liquid composition at each plate. Thus an Mth order system represented an M plate absorption column. All input data were generated off-line on a large computer and the estimates produced on the PDP 8 were compared with those generated off-line. No significant difference was apparent, as is described in the Appendix.

Discussion

Industrial processes often have large time constants and a sample interval of 1 minute would be unnecessarily short

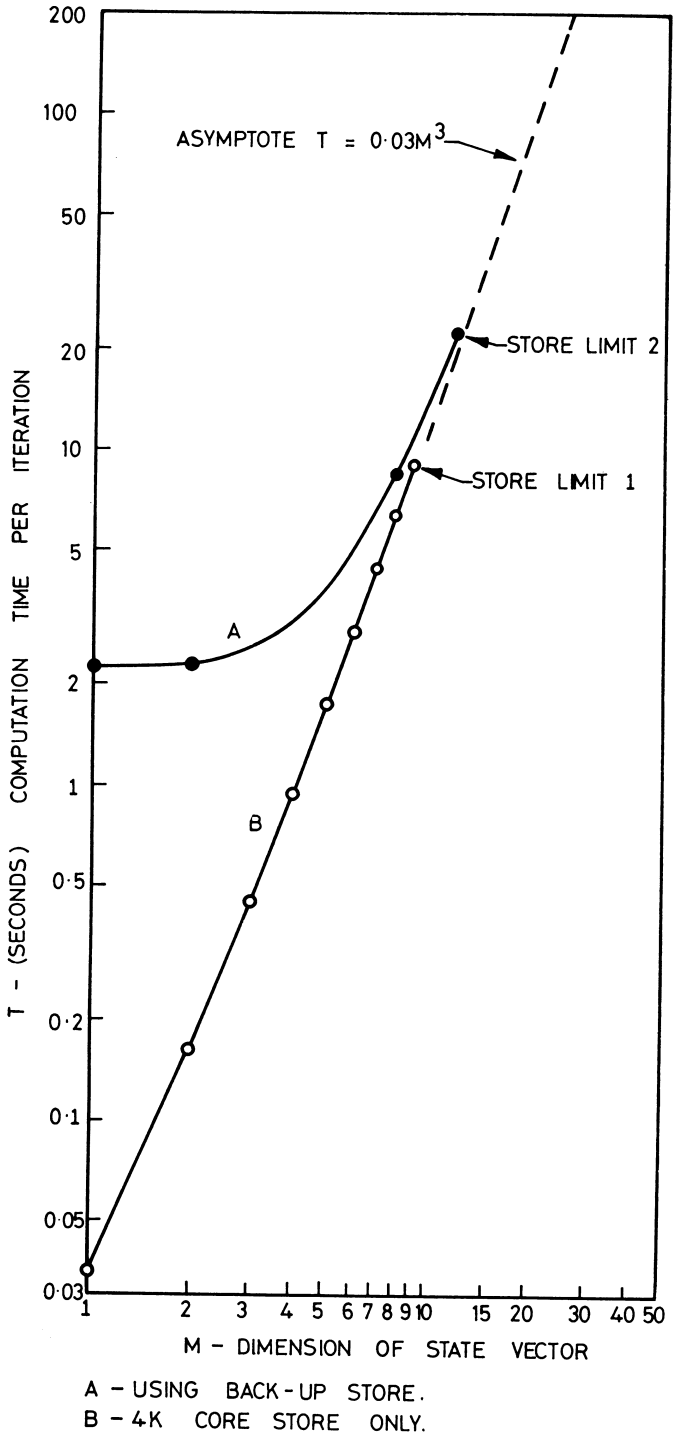

Fig. 1. PDP-8 Computation Time vs. Size of System

for many applications whilst in a few cases 20 minutes might be adequate for optimisation purposes. Thus the results in Fig. 1 show that the PDP 8, even in its basic 4K form, would be quite satisfactory for on-line state estimation of linearised systems that can be described by 9 state variables or 12 variables using back-up store. With certain approximations these might include: up to three reactors in series; polymerisation reactions; separation processes, e.g. distillation, evaporation, gas absorption; or blending systems.

In exceptional cases estimation of as many as 50 state and parameter values may be required and we can extrapolate the results to determine whether this is feasible. A computer with 28K (12 bit) core and magnetic tape back-up would have adequate storage capacity for estimation and control software but computation time would be about 1 hour. With hardware multiply/divide facilities this might be reduced to 20 to 30 minutes and on a machine requiring less than three words for a floating point number a much smaller core store would be needed. It would seem that on-line state estimation of a 50th order system is already within the realms of possibility.

The estimation algorithm may be extended to provide local rather than global linearisation of non-linear systems by matrix exponentiation. This together with a non-diagonal measurement matrix **H** and time-variant statistical characteristics of the disturbances would increase significantly both computation time and storage space required. However, we can trade accuracy for speed in several ways, for example by reducing the order of the approximation to the real system, and still produce estimates of key variables which are good enough to justify the use of the estimator. On the other hand, future computer installations on large process plants are likely to comprise one or more small machines, which simply mind the plant, linked to a remote time-shared computer which is capable of large scale arithmetic.

Conclusion

We have shown that on-line sequential state (and parameter) estimation is feasible in terms of both speed and accuracy with computers of the size used for the control of industrial processes. The possibilities arising extend from control based on unmeasured variables through to the application of modern control theory to noisy non-linear systems.

Further studies are to include applications on real processes.

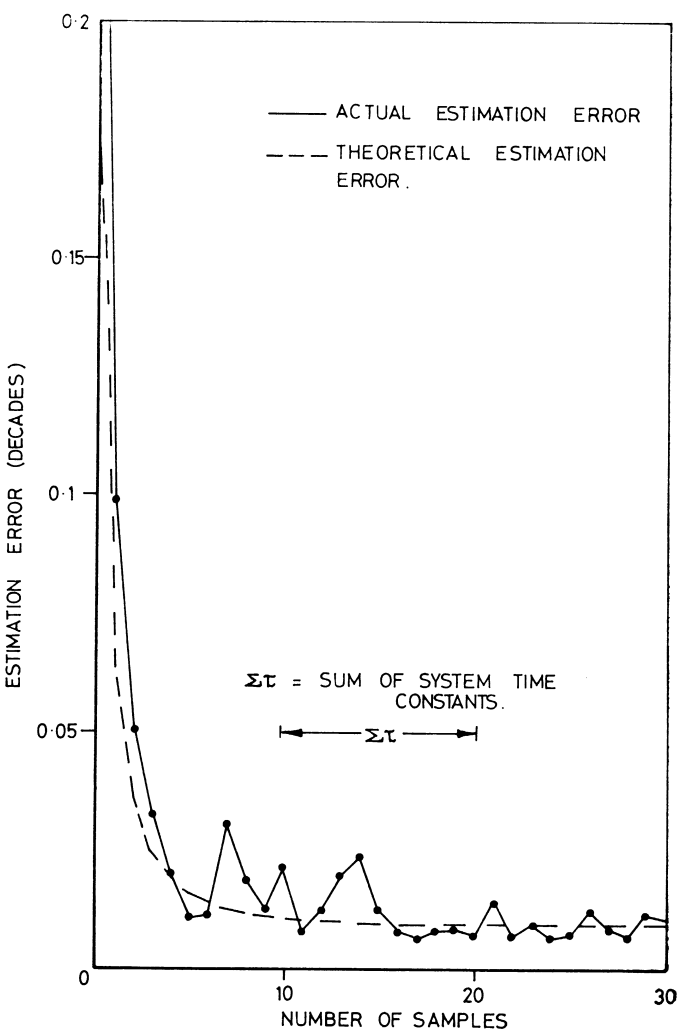

Fig. 2. Estimation Error vs. Number of Samples

Acknowledgement

The authors are grateful to Esso Petroleum Company Limited for supporting this investigation.

Appendix

Convergence and accuracy of the filter

In order to evaluate the performance of the filter it is convenient to express the overall error of estimation in a single value. When using a digital simulation to generate 'plant' data we know the actual state x_a and can calculate the estimation error for any variable as

$$d(k) = \log_{10}(x_f(k)/x_a(k)) \text{ decades}$$

as used by Coggan and Noton (1969). In any real application x_a is never known. The **G** matrix of error covariances for the estimates is all the information we have. For any variable we can formulate a theoretical estimation error as

$$d(k) = \log_{10}(1 + \sigma_f(k)/\bar{x}) \text{ decades}$$

We can take the RMS value of $d(k)$ over the M states and arrive at an overall estimate of the accuracy of estimation. In the case of digital simulation both actual and theoretical errors can be calculated and compared. If the filter is operating correctly the two values should agree fairly closely, especially after convergence of the filter.

A typical set of estimation errors is shown in **Fig. 2.** There is reasonable agreement towards the end of the range and convergence is rapid. In this case the system was an eight stage gas absorber in unsteady state taking four measurements of composition at each sample. Random process disturbances and measurement errors had a standard deviation of 10% (0·04 decades) of the nominal state ($\bar{x}$). It should be noted that such models are used for demonstration purposes only and are not primarily intended as examples of industrial applications.

The calculations were conducted on a large machine, retaining 12 figures for floating point arithmetic, and on the PDP 8, which retains seven figures. The same algorithm was used in both cases. Six figure agreement between the two machines was obtained for x_f. The filter matrix (**F**) agreed to six figures on the diagonal and to five figures elsewhere.

References

BUCY, R. S., and JOSEPH, P. D. (1968). Filtering for stochastic processes with applications to guidance, *Interscience*, New York.

COGGAN, G. C., and NOTON, A. R. M. (1969). Discrete-time sequential state and parameter estimation in chemical engineering, I.Chem.E. Symposium entitled 'Current trends with computers in chemical engineering', *Trans. Instn. Chem. Engrs.*, Vol. 48, pp. T245-T264

KALMAN, R. E. (1960). A new approach to linear filtering and prediction problems, *Trans. ASME*, Series D, Journal of Basic Engineering, Vol. 82, pp. 35-45.

Correspondence

To the Editor
The Computer Journal

(Note on)² Algorithm 44

Sir,

Broyden's reply to Fielding's 'Note on Algorithm 44' (Fielding, 1970; Broyden, 1970) would certainly not get my blessing. In fact, I would not accept his excuse for the programming errors pointed out by Fielding from the lowliest of the 2,000-odd students I have taught to program (which is pretty lowly—would you believe a 10-year-old?).

However, he does raise two questions:

(*a*) Is the use of a declared variable in an arithmetic expression *before* it has been assigned a value legal in ALGOL 60?

A lawyer would call this a 'nice' point. The answer is not set out clearly in the Report (Naur, 1963). In Section 3.3.3. Semantics (of Arithmetic Expressions) operations are required to operate on the actual numerical values of the primaries of the expression. For a variable the actual value is defined as . . . 'the current value (assigned last in the dynamic sense) . . .'. Declaration of a variable, Section 5.1.3. does not mention assignment of a value so that we can argue that no 'current value' exists and hence no actual value and hence that we cannot execute an operation on it. I will leave it to the ALGOL theologians to decide whether that is an acceptable argument—and if it applies even to *theta* 2: = *theta* 1;. If the argument is not valid, it seems a gross oversight of the ALGOL 60 language.

It is worth pointing out that the much maligned (by ALGOL enthusiasts) ANSI FORTRAN (1966) recognises the distinction between 'declaration' and 'assignment' ('definition' in the Standard) and explicitly forbids use of a variable before its value has been 'defined'. Of course, not all the compilers that claim to implement FORTRAN will get this right—I would not bet on the Leeds compiler for example (Wells, 1970; Hammersley and Larmouth, 1970).

(*b*) Whatever the legal situation what should a processor do?

I would maintain very strongly that it should *always* detect—and forbid—the usage of an undefined value. When the hardware, of a processor can detect certain illegal bit patterns, this is not too expensive. For example, the IBM 7040 detects and traps the use of a word with bad parity. The authors of 7040 WATFOR (Shantz *et al.*, 1966) discovered a way of deliberately setting up such words and of disabling the trap so that they had a convenient and economic solution to the problem.

When the hardware facility is not available a valid value must be used. Zero has often been a choice and at least gives consistent results—important when the hardware is unreliable. There is a greater chance of picking up the error if the default value is the largest real number representable in the processor (or what about the current time of day?). In any case there should be at least a program testing mode that optionally does a rigorous test—and of course an optimising compiler could pick up many cases during its flow analysis.

Yours faithfully,

K. A. REDISH

Department of Applied Mathematics
McMaster University
Hamilton, Ontario, Canada
10 July 1970

References

American National Standards Institute, Standard X 3.9—1966—FORTRAN.

BROYDEN, C. G. (1970). Algorithm 44, *The Computer Journal*, Vol. 12, p. 406.

FIELDING, K. (1970). Note on Algorithm 44, *The Computer Journal*, Vol. 13, p. 219.

HAMMERSLEY, P., and LARMOUTH, J. (1970). Letter Towards FORTRAN VI, *The Computer Journal*, Vol. 13, p. 220.

NAUR, P. (ed.) (1963). Revised Report on the Algorithmic Language ALGOL 60, *The Computer Journal*, Vol. 5, p. 349.

SHANTZ, P. W. *et al.* (1966). WATFOR Documentation. Comp. Sc. Dept., University of Waterloo.

WELLS, M. (1970). Letter, Towards FORTRAN VI, *The Computer Journal*, Vol. 13, p. 120.