# Real Time Computing III Implementing Linear Filtering and Control Algorithms

Källström, Claes; Åström, Karl Johan

1971

*Document Version:*
Publisher's PDF, also known as Version of record

Link to publication

*Citation for published version (APA):*
Källström, C., & Åström, K. J. (1971). *Real Time Computing III Implementing Linear Filtering and Control Algorithms*. (Research Reports TFRT-3040). Department of Automatic Control, Lund Institute of Technology (LTH).

*Total number of authors:*
2

# REAL TIME COMPUTING III IMPLEMENTING LINEAR FILTERING AND CONTROL ALGORITHMS.

C. KÄLLSTRÖM
K. J. ÅSTRÖM

REPORT 7122(B) DECEMBER 1971
LUND INSTITUTE OF TECHNOLOGY
DIVISION OF AUTOMATIC CONTROL.

# REAL TIME COMPUTING III
# IMPLEMENTING LINEAR FILTERING AND CONTROL ALGORITHMS.

K.J. Åström

C. Källström

ABSTRACT.

The report covers some practical aspects on the implementation of Kalman filters and linear regulators on small computers. Particular attention is given to synchronization of inputs and outputs, computing time and memory requirements. The algorithms discussed are general purpose FORTRAN algorithms, using floating point arithmetic. Special techniques like processing measurements one-by-one, square root representation of the covariance matrix etc. are not discussed.

This work has been supported by the Swedish Board for Technical Development under Contract 71-50/U33.

TABLE OF CONTENTS

## 1. INTRODUCTION.

The importance of software for process control and other real time application is increasing rapidly. There are many reasons for this, e.g.

- The cost of hardware is decreasing rapidly while the performance of hardware is increasing.
- The comparatively simple algorithms (like PID) are successively being replaced by more sophisticated algorithms.
- Progress in control theory is leading to new algorithms.

In a university environment it is important that students are confronted with real problems as quickly as possible. This can be done in our real time control and computing laboratory. The efficient use of the facilities requires, however, that a reasonable number of well-tested algorithms are available. To achieve flexibility the algorithms should also be general purpose and written in a high level language which for practical reasons has been chosen as FORTRAN. From an educational point of view it is also a requirement that the algorithms meet reasonable software requirements.

This report is one of a sequence which deals with real time computations. The report is devoted to Kalman filtering and linear control algorithms. It is hoped that the results can also be of interest to industry. They can at least be used for feasibility studies to estimate computational load and memory requirements. The actual code is also written in such a way that it should be easy to transport to different computers. The algorithms are actually implemented on the PDP 15/35. To achieve flexibility we have not exploited any tricks like processing measurements one-by-one, exploiting a particular structure, using a triangular representation of the covariance matrix etc.

The synchronization of inputs and outputs is discussed in Section 2. The equations for the linear regulator algorithm are discussed in Section 3.

The implementation of the linear quadratic regulator in the case of precomputed gain is also discussed in this section. It is found that the algorithm requires a code of 256 cells. The data storage for a system with 5 inputs, 5 outputs and 10 states is 704 cells of which 64 is a common area SLASK. The computing time for one step is 64 m sec.

Organization of the calculations is discussed in Section 4. Particular attention is given to the order in which operations have to be performed, the increase and decrease of the data base during the computations and the possibilities to interrupt the algorithm. Storage and computing times are also given in Section 4. It is shown that one iteration of the algorithm including the solution of the time varying Riccati equation can be done in 1.2 - 1.7 seconds for a tenth order system with 5 inputs and 5 outputs. The code for the regulator algorithm itself requires about 700 cells. If all necessary subroutines like matrix inversion, real and integer arithmetic and diagnosis are included the algorithm requires about 3000 cells. The data storage for a system with 10 states, 5 inputs and 5 outputs is about 2 k of which 1 k is a common dummy area SLASK.

## 2. SYNCHRONIZATION OF INPUTS AND OUTPUTS.

When implementing any control algorithm on a digital computer many variables are by necessity discontinuous. To avoid confusion it is wise to decide once and for all if the variables are continuous from the right or from the left. In this note we adopt the convention that all variables are continuous from the right. See Fig. 2.1.

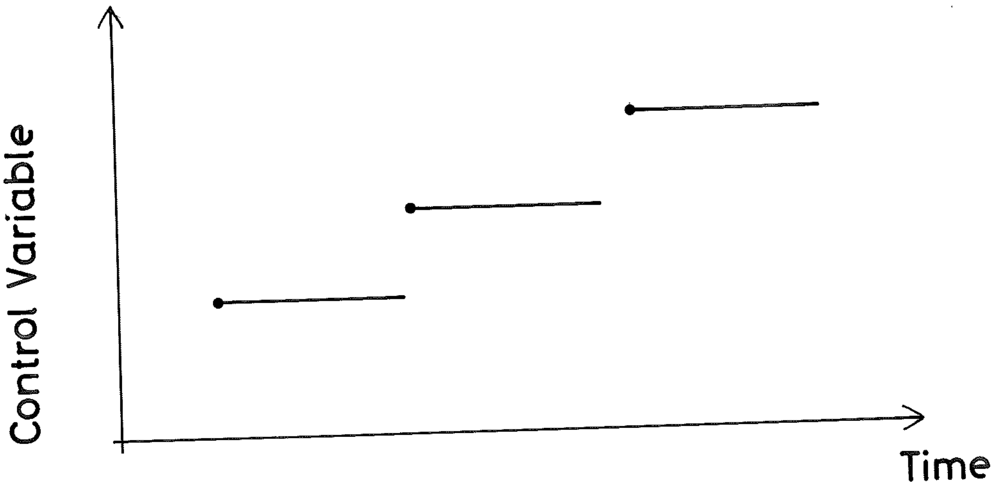

Fig. 2.1 - Control and measured signals for a typical process control system. Notice that the control signal is discontinuous.

Another detail that often causes confusion is the definition of inputs and outputs. Viewed from the process control variables like value settings are naturally considered as inputs while the measured variables are regarded as outputs. From the regulators (computers) point of view it is, however, natural to consider the measured variables as inputs and the control variables as outputs. We will therefore mostly use the words measured signal y and control signal u to avoid confusion. When we use input and output we will refer this to the process, i.e. input ≡ control variable u and output ≡ measured variable y.

There are basically two different ways to implement the control algorithms:

- Case A. In this situation the measured variables y are read at time t and the control variable $u(t+1)$ to be set at time $t+1$ is computed from $y(t)$.
- Case B. The measured variables are read at time t and the control variables are evaluated as quickly as possible and set at time $\underline{\tau}$ where $\underline{\tau}$ is the smallest time required to do the computations.

Case A has the disadvantage that the control actions are delayed unnecessarily and case B has the disadvantage that the delay will be variable depending upon the program. Changes in the program and in the priorities will thus result in a variable delay. If the delay is critical this means that the result will be sensitive to changes in programming.

Notice that the software requirements are different in the two cases.In case A the order of the computations is not crucial while in case B the computations should be arranged in such a way that the amount of computations required to compute the control variable from the output is as small as possible. Updating of state variables and intermediate results are then conveniently done after the control variable is set.

Another practical detail is that there is a good rule to read the inputs before the outputs are set out. If this is not done there is always the risk of electrical cross coupling.

Also notice that unless the computational lag is very small it is necessary to derive special equations in case B since the regular sampled data theory assumes that outputs are changed at the sampling instants.

The sequence of events in the different cases will now be considered in more detail.

## Case A.

The sequence of events is as follows:

1. Read measured variable $y(t)$.
2. Set control variable $u(t)$ computed during previous sampling interval.
3. Compute the control variable $u(t+1)$ to be executed at time $t+1$ based on past data and the measurement $y(t)$.
4. Wait.

This case is illustrated in Fig. 2.2.

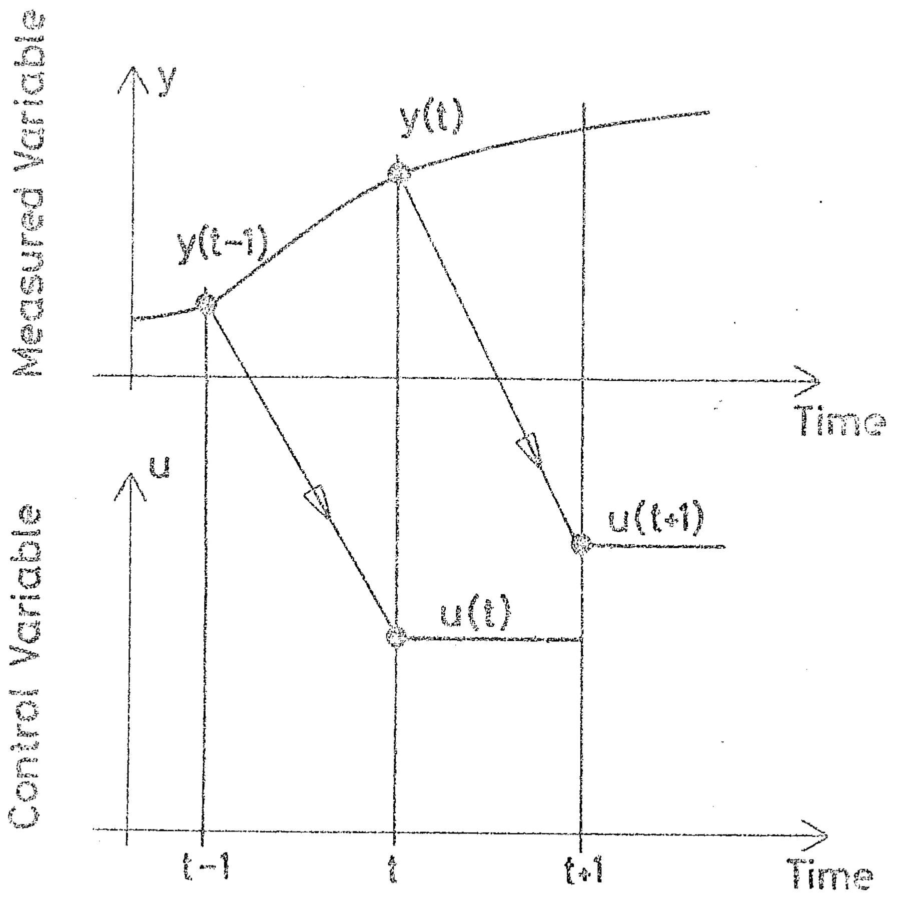

Fig. 2.2 - Illustration of the synchronization of inputs and outputs in case A.

Case B.

In the case B the sequence of events are:

1. Read measured variable y(t).
2. Compute control variable u(t), i.e. the output to be executed in the interval (t,t+1).
3. Set the control variable u(t) at time $t+\tau$.
4. Update the necessary variables.
5. Wait.

See Fig. 2.3.

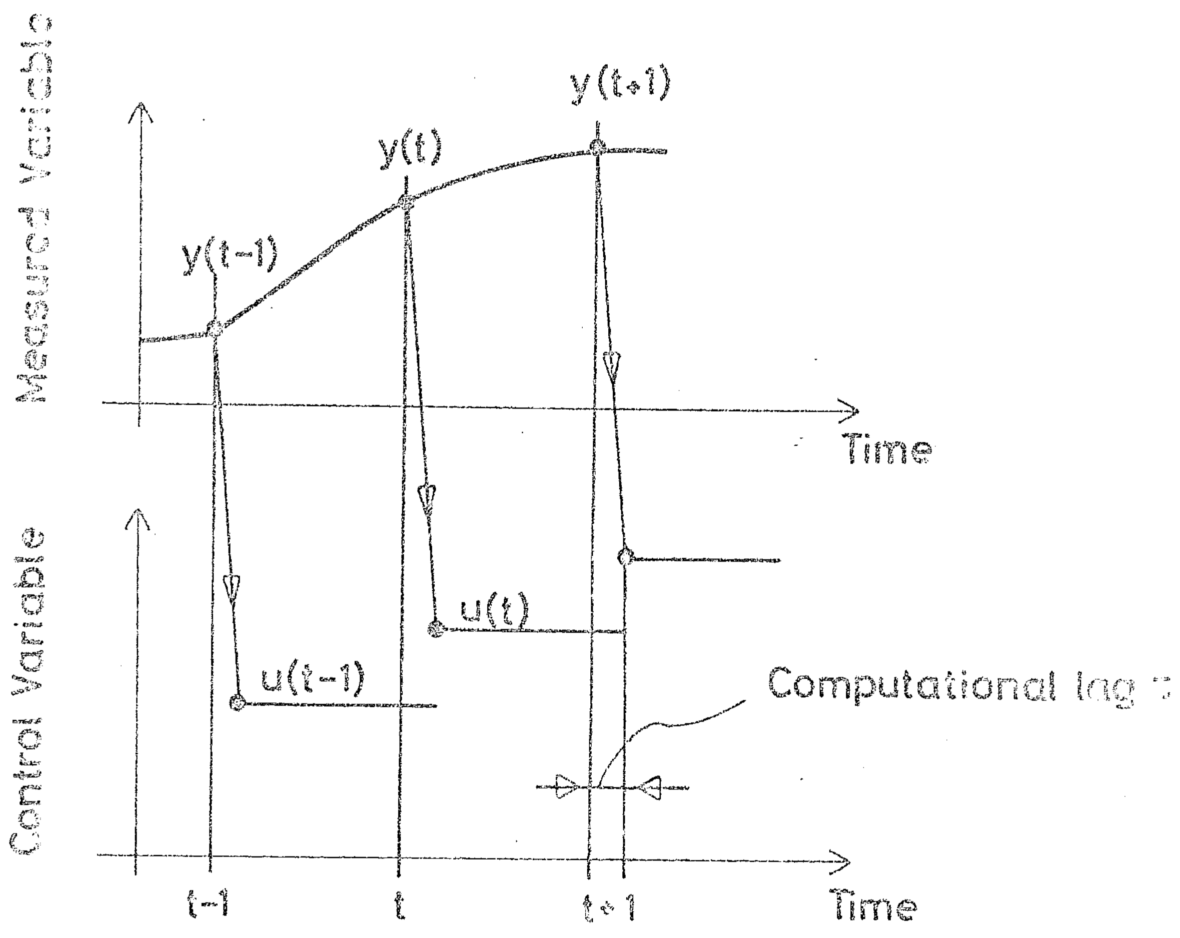

Fig. 2.3 - Illustration of synchronization of inputs and outputs in case B.

When making laboratory experiments it is highly desirable to have general software in order to avoid too much programming. In such a case there are only two cases that are of interest, namely case A and the specialization of case B when the computing lag τ can be neglected. In the following we will only discuss case A.

## 3. THE LINEAR REGULATOR ALGORITHM.

The sequence of events in case A was discussed in the previous section. The appropriate equations for the linear regulator which are valid in this case are given in [1]. We have:

$$\hat{x}(t+1|t) = \Phi(t+1|t)\hat{x}(t|t-1) + \Gamma(t)u(t) + K(t)[y(t) - \hat{y}(t|t-1)] \tag{3.1}$$

$$\hat{y}(t|t-1) = \theta(t)\hat{x}(t|t-1) \tag{3.2}$$

$$u(t+1) = - L(t+1)\hat{x}(t+1|t) \tag{3.3}$$

$$K(t) = [\Phi(t+1;t)P(t)\theta^T(t) + R_{12}(t)] \cdot [\theta(t)P(t)\theta^T(t) + R_2(t)]^{-1} \tag{3.4}$$

$$P(t+1) = \Phi(t+1;t)P(t)\Psi^T(t+1;t) + R_1(t) - R_{12}(t)K^T(t) \tag{3.5}$$

$$\Psi(t+1;t) = \Phi(t+1;t) - K(t)\theta(t) \tag{3.6}$$

We will now discuss different ways to organize the computations described by these equations.

## 3.1. Precomputed Filter Gain 1.

We first observe that the filter gains can be precomputed. If this is done we see that in order to do the computations it is necessary to have a data set containing the system parameters

$\Phi(t+1;t)$, $\Gamma(t)$, $\theta(t)$, $K(t)$, $L(t+1)$

the measurement

$y(t)$

the control variable

$u(t)$

and the state variable

$\hat{x}(t|t-1)$

The calculations thus require a data set of

$$N = n_x(n_x+2n_u+2n_y+1) + n_u + n_y$$

parameters.

The equations can be implemented by the algorithms GLIRE1 and GLISY1 given in Appendices A and B. The subroutines have been compiled on PDP 15/35 by the FORTRAN V12D compiler and require 116 resp. 140 cells. Computing times are shown in Appendix C. Cf. ref. [2].

## 3.2. Precomputed Filter Gain 2.

If the gain K is precomputed additional savings in storage and computing time can be obtained if it is observed that the equations (3.1), (3.2) and (3.3) can be rewritten as

$$\hat{x}(t+1|t) = [\Phi(t+1;t) - K(t)\theta(t)]\hat{x}(t|t-1) + $$
$$+[\Gamma(t) \quad K(t)]\begin{vmatrix} u(t) \\ y(t) \end{vmatrix} \qquad (3.7)$$

$$u(t+1) = - L(t+1)\hat{x}(t+1|t) \qquad (3.8)$$

Hence if $\Psi(t+1;t) = \Phi(t+1;t) - K(t)\theta(t)$ is precomputed it is only necessary to have a data set consisting of the system parameters

$\Psi(t+1;t)$, $\Gamma(t)$, $K(t)$, $L(t+1)$

and

$\hat{x}(t|t-1)$, $y(t)$, $u(t)$

The computations thus require a data set consisting of

$$N = n_x(n_x+2n_u+n_y+1) + n_u + n_y$$

parameters.

The equations can be implemented with the algorithm GLISY1 previously given.

## 3.3. Real Time Computation of Filter Gain.

The filter gain can also be computed in real time as the process develops. In comparison with the previous schemes this is in essence a trade off between computing time and storage. The equation (3.4) implies that to compute the filter gain in real time it is also necessary to compute the covariance P of the estimation error in real time. This will give some additional advantages because in each step of the calculations there is an estimate of the covariance of the measurement error available, namely

$$R_y = \theta(t)P(t)\theta^T(t) + R_2(t) \tag{3.9}$$

This implies that in each step of the calculations it is possible to test if the measurement y(t) is reasonably close to its estimated value $\hat{y}(t|t-1)$. Under the Gaussian assumption the conditional distribution of y(t) given data observed up to time t-1 is thus Gaussian with mean $\hat{y}(t|t-1)$ and the covariance $R_y$ given by (3.9).

It is then possible to answer several statistical problems. Two cases will be considered. In the first case it is assumed that the most likely failure mechanism is that a break-down in the whole transmission channel for the measurements or an A/D failure is the most likely error. This means that the crucial problem is to accept or reject the whole measuremet vector. In the other case the individual measurements will be accepted or rejected.

<u>All measurements accepted or rejected.</u>

The random variable

$$t = e^T R_y^{-1} e \qquad (3.10)$$

where

$$e = y(t) - \hat{y}(t|t-1) \qquad (3.11)$$

is thus $\chi^2$ with $n_y$ degrees of freedom. Suitable test levels are found in tables over the $\chi^2$ distribution.

If a measurement is rejected the equations for updating the state estimate and the covariance should be changed to

$$\hat{x}(t+1|t) = \Phi(t+1;t)\hat{x}(t|t-1) + \Gamma(t)u(t) \qquad (3.12)$$

$$P(t+1) = \Phi(t+1;t)P(t)\Phi^T(t+1;t) + R_1(t) \qquad (3.13)$$

If the filter gain is computed in real time is thus necessary to have a data set consisting of the system parameters

$$\Phi(t+1;t),\ \Gamma(t),\ \theta(t),\ L(t+1),\ R_1(t),\ R_{12}(t),\ R_2(t)$$

and the

$$P(t),\ K(t),\ \hat{x}(t|t-1),\ u(t),\ y(t)$$

A data set of

$$N_1 = n_x[3n_x+2n_u+3n_y+1] + n_u + n_y^2 + n_y \qquad (3.14)$$

parameters is thus required. If we consider that the covariance matrices $P(t)$, $R_1(t)$ and $R_2(t)$ are symmetric, only a data set of

$$N_2 = n_x[2n_x+2n_u+3n_y+2] + n_u + \frac{1}{2} n_y[n_y+3] \qquad (3.15)$$

parameters is required. The increase in the data set compared with precomputed filter gain 2 is thus

$$\Delta N_2 = n_x[n_x+2n_y+1] + \frac{1}{2} n_y[n_y+1]$$

parameters.

<u>Individual measurements accepted or rejected</u>.

If the disturbances and the initial state are Gaussian the conditional probability distribution of the outcome of a single measurement is normal with mean value $y_i(t|t-1)$ and covariance $(R_y)_{ii}$ where $(R_y)_{ii}$ is the ii:th element of the matrix $R_y$ given by (3.9). A suitable test quantity is thus

$$t_i = e_i/\sqrt{(R_y)_{ii}} \qquad i = 1, 2, \ldots, n_y \qquad (3.16)$$

which is normal (0,1). At each step of the iteration all test quantities $\{t_i, i = 1, 2, \ldots, n_y\}$ are thus evaluated. The filter gain $K(t)$ is then computed according to (3.4), where the matrices $\theta(t)$, $R_{12}(t)$ and $R_2(t)$ have been reduced to consider only the accepted measurements. The columns of $K(t)$ corresponding to rejected measurements are then put zero. This is faster than double indexing!

## 4. ORGANIZATION OF THE PROGRAM FOR THE TIME VARYING FILTER.

Knowing the algorithms we will now discuss the organization of the code.

### Requirements.

Before going into the details we will first state some general requirements.

- o The program will typically be used in an environment where several filters are implemented in the same computer. It is therefore desirable to have a code which can be shared among many loops.
- o The data base (state) of the algorithm will grow and shrink during the computations. To avoid unnecessarily large storage we are willing to compromize by making the code nonreentrant. The main gain in storage is obtained by putting three 13x13 matrices in the common block SLASK.
- o The algorithm will be implemented on the PDP 15/35. It should be easy to transport the algorithm to other computers. We are, however, willing to make provisions that the FORTRAN code is compiled efficiently on the PDP 15/35.
- o The numerical properties of the discrete Riccati equation are not sufficiently well explored. Reasonable error estimates are not available. There are only two critical parts in the algorithm, solution of a system of linear equations and taking the difference between matrices. The linear equation solver can be checked by testing the pivot element. It is desirable to have access of this

test quantity outside the algorithm. Another possibility to test the overall result is to check the matrix P for symmetry. This will, however, require additional computing time and storage.

## Subroutines or inline code.

The major computational load is to solve the Riccati equations (3.5), (3.6) and to compute the gain K given by (3.4). The computations involved are matrix additions and multiplications and solution of linear equations. One problem is then if these operations should be made as subroutines or if they should be coded in-line. The trade-offs will naturally be machine dependent. The following analysis will be based on the FORTRAN V12D compiler of PDP 15/35. Since all arguments are given in detail it is, however, easy to find the changes if other compilers are used. It is also believed that the results are typical for many medium sized computers.

The FORTRAN statements for matrix additon

```
     DO 10 I=1,N
     DO 10 J=1,M
10   C(I,J) = A(I,J)+B(I,J)
```

is compiled in to a code consisting of 37 cells. The subroutine

```
     SUBROUTINE MAD (A,B,C,N,M)
     DIMENSION A(1,1), B(1,1), C(1,1)
     DO 10 I=1,N
     DO 10 J=1,M
10   C(I,J) = A(I,J) + B(I,J)
     RETURN
     END
```

is compiled in a code consisting of 54 cells. The subroutine call

```
    CALL MAD (A,B,C,N,M)
```

is compiled in a code consisting of $2+2\times5 = 12$ cells. We thus find that from the point of view of minimizing the code it is advantageous to use a matrix addition subroutine if more than two matrix additions are made in the program. For more complicated operations like matrix multiplication and solution of linear equations where the code is very much longer than the code for a subroutine call the critical limit is much lower.

## Timing considerations.

We have thus found that from the point of view of storage it is often advantageous to use subroutines. However, the computing time will obviously increase when inline code is replaced by subroutines and subroutine calls. We thus have an example of the typical trade-off between storage and computing time. To find the orders of magnitude involved we will again consider the matrix addition. The time consuming part of the matrix addition are the following operations:

| | | | |
|---|---|---|---|
| JMS× | .SS | Find address of A(I,J) | 100 μs |
| JMS× | .SS | Find address of B(I,J) | 100 μs |
| JMS× | .SS | Find address of C(I,J) | 100 μs |
| JMS× | .AG | Floating load A | 40 μs |
| JMS× | .AI | Floating add | 200 μs |
| JMS× | .AH | Floating store C | 40 μs |

The major loop of the matrix addition code thus requires at least 580 μs.

A subroutine call requires 42+18×(NUMBER OF ARGUMENTS) μs which in the particular case means 132 μs. We thus find that the increase in time is very modest (only 1.2% increase for 5×5 matrices and 0.2% for 10×10 matrices). Notice, however, that the trade-off may be different for a computer with floating point hardware.

In summary we thus find that by introducing a matrix addition subroutine a saving in memory is made if more than three additions are made and that the increase in computing time is modest. For more complex operations like matrix addition and solution of linear equations the savings are even greater. We can thus conclude that it seems reasonable to use subroutines for the matrix operation. Notice, however, that we can loose a bit because operations like D=A×B+C can be coded efficiently like

```
      DO 10 I=1,N
      DO 10 J=1,M
      R=C(I,J)
      DO 11 K=1,L
11    R=R+A(I,L)×B(L,J)
10    D(I,J)=R
```

while it requires two subroutine calls if matrix add and matrix multiply are the only available matrix routines.

## Four different cases.

Having found that it is advantageous to use subroutines we will now procede to discuss more details of the code. We have previously found that two cases are of interest namely when the whole measurement vector is accepted or rejected and when the individual measurements are accepted and rejected. Since many expressions are symmetric it is also of interest to see if this can be explored. Notice, however, that the use of symmetry means that ordinary matrix routines cannot be used. We will thus quite arbitrarily consider four cases.

1. All measurements accepted or rejected. Symmetry is not exploited. RUDY.
2. Same as 1 but symmetry is exploited. RUDY2.
3. The individual measurements accepted or rejected. Symmetry is not exploited. RUDY1.
4. Same as 3 but symmetry is exploited. RUDY3.

## RUDY.

In the case that the whole measurement vector is accepted or rejected and the symmetry is not exploited the code can be organized as follows:

1. Enter system parameters to be used in the computations done during the interval $(t,t+1)$, i.e. $\Phi(t+1;t)$, $\Gamma(t)$, $\theta(t)$, $R_1(t)$, $R_{12}(t)$, $R_2(t)$ and $L(t+1)$. (Notice $L(t+1)$!)
2. Enter the measurement $y(t)$, the control signal $u(t)$, and the state $\hat{x}(t|t-1)$, $P(t)$.

3. Compute the residuals

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - \theta(t)\hat{x}(t|t-1)$$

4. Compute the covariance of residuals

$$R_y = \theta(t)P(t)\theta^T(t) + R_2(t)$$

5. Compute

$$t = e^T(t)\, R_y^{-1}\, e(t)$$

and test if t is larger than the test level. Let the indicator IND = 1 if measurement rejected, and IND = 0 if measurement accepted. IND = 2 if $R_y$ singular or close to singular or if decomposition impossible.

6. Compute filter gain

$$K(t) = [\Phi(t+1;t)P(t)\theta^T(t) + R_{12}(t)]R_y^{-1}$$

7. Update state estimate

$$\hat{x}(t+1|t) = \Phi(t+1;t)\hat{x}(t|t-1) + \Gamma(t)u(t) + \begin{cases} K(t)e(t) \\ \text{if IND=0} \\ 0 \\ \text{if IND=1} \end{cases}$$

8. Compute output to be used during the next interval

$$u(t+1) = - L(t+1)\hat{x}(t+1|t)$$

9. Update covariance of state estimate

$$P(t+1) = \begin{cases} \Phi(t+1;t)P(t)\Psi^T(t+1;t) + R_1(t) - \\ - R_{12}(t)K^T(t) \qquad \text{if IND} = 1 \\ \Phi(t+1;t)P(t)\Phi^T(t+1;t) + R_1(t) \\ \qquad\qquad \text{if IND} = 0 \end{cases}$$

$$\Psi(t+1;t) = \Phi(t+1;t) - K(t)\theta(t)$$

Notice that there are certain restrictions in the order in which the computations can be done. The step 9 can, however, be done before step 7.

Notice the way in which the data base shrinks and grows. To avoid repetition of calculation it is desirable to store $R_y$ from step 4 or rather its triangular decomposition since it will be used also in 6 to compute K. We have the choice of storing either $\Phi P$ or $\theta P$ which appear in steps 4, 6 and 9. Since the computation of $\Phi P$ is more time consuming we decide to stor $\Phi P$. This quantity is computed in step 6 and it is also needed in step 9. It is also necessary to store K obtained in step 6 to avoid repeating the calculation of K in step 9. We thus find that in order to avoid unnecessary repetition of computations it is necessary to store three matrices. An analysis of step 9 also shows that this step can be computed without additional storage. Notice that these arguments are based on the assumption that computing time is more important than storage. Notice, however, that if it is assumed that K is stored it is not possible to evaluate step 9 with less than two additional dummy matrices. We can thus conclude that three dummy matrices are required in order to carry out the calculations. This number makes it possible to do the calculations efficiently and it cannot be decreased even if we are willing to sacrifice computing time.

In our PDP 15/35 we have a common block SLASK for dummy matrices consisting of 1024 cells. The maximum system order that can be used is thus $n_x = 13$. With this number 1014 cells of the common block slask is used. The actual code for the algorithm is now straightforward. See the subroutine RUDY in Appendix D. On the FORTRAN V12D compiler the FORTRAN program compiles to a code consisting of 721 cells. This includes the subroutine MADD which consists of 83 cells. If all subroutines required are included the code requires a total of 2993 memory locations. A list of the storage locations required by the different routines is shown in Table 4.1. The data storage depends on the dimensions of the system. The number of parameters are given by (3.14).

| Subroutine | | Number of cells |
|---|---|---|
| RUDY | | 638 |
| DESYM | Triangular Decomposition Sym. Matr. | 497 |
| SOLVS | Solution of triangular equation | 297 |
| NORM | Matrix norm | 134 |
| MMULT | Matrix multiply | 112 |
| SCAPRO | Scalar product | 265 |
| ABS | Absolute value | 14 |
| SQRT | Square roat | 58 |
| RELEAE | Real arithmetic | 546 |
| INTEAE | Integer arithmetic | 76 |
| DOTSS | Address to indexed variables | 74 |
| SPMSG | Stop and pause messages | 59 |
| OTSER | Object time system error | 60 |
| MOVE | Move an array | 25 |
| .DA | Address of argument in subroutine jump | 39 |
| .CB | Normalize | 16 |
| MADD | Matrix add and subtract | 83 |
| | | 2993 |

Table 4.1 - The number of memory locations required for the subroutine RUDY and all system and library routines.

The total code for the algorithm thus requires 2993 cells. The number of cells required to store the data will depend on the size of the matrices. The number of elements to be stored are given in Section 3. Some examples are given in Table 4.2. It is also necessary to have 1024 cells for the common block SLASK. It is, of course, possible to decrease the size of the common block SLASK if the system order $n_x$ is less than 13 and increase the size of $n_x$ greater than 13.

Examples of execution times are given in Table 4.3. The number of control variables $n_u$ is one, but the influence of $n_u$ can be neglected. Note that the average addition and multi.oication time is 200 μs.

| $n_x$ \ $n_y$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 24 | | | | | | | | | | | | |
| 2 | 54 | 84 | | | | | | | | | | | |
| 3 | 96 | 136 | 180 | | | | | | | | | | |
| 4 | 150 | 200 | 254 | 312 | | | | | | | | | |
| 5 | 216 | 276 | 340 | 408 | 480 | | | | | | | | |
| 6 | 294 | 364 | 438 | 516 | 598 | 684 | | | | | | | |
| 7 | 384 | 464 | 548 | 636 | 728 | 824 | 924 | | | | | | |
| 8 | 486 | 576 | 670 | 768 | 820 | 976 | 1086 | 1200 | | | | | |
| 9 | 600 | 700 | 804 | 912 | 1024 | 1140 | 1260 | 1384 | 1512 | | | | |
| 10 | 726 | 836 | 950 | 1068 | 1190 | 1316 | 1446 | 1580 | 1718 | 1860 | | | |
| 11 | 864 | 984 | 1108 | 1236 | 1368 | 1504 | 1644 | 1788 | 1936 | 2088 | 2244 | | |
| 12 | 994 | 1124 | 1258 | 1396 | 1538 | 1684 | 1834 | 1988 | 2146 | 2308 | 2474 | 2644 | |
| 13 | 1176 | 1316 | 1460 | 1608 | 1760 | 1916 | 2076 | 2240 | 2408 | 2580 | 2756 | 2936 | 3120 |

Table 4.2 - The number of cells $N = 2[n_x(3n_x+2n_u+3n_y+1) + n_u + n_y(n_y+1)]$ required to store the data for RUDY, RUDY1, RUDY2 and RUDY3. It is assumed that $n_u=n_y$.

| $n_x \downarrow$ / $n_y \rightarrow$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0.05 | 0.08 | | | | | | | | | | | |
| 3 | 0.10 | 0.14 | 0.18 | | | | | | | | | | |
| 4 | 0.17 | 0.22 | 0.27 | 0.33 | | | | | | | | | |
| 5 | 0.26 | 0.32 | 0.38 | 0.45 | 0.53 | | | | | | | | |
| 6 | 0.36 | 0.43 | 0.51 | 0.59 | 0.69 | 0.80 | | | | | | | |
| 7 | 0.50 | 0.59 | 0.68 | 0.78 | 0.89 | 1.01 | 1.15 | | | | | | |
| 8 | 0.68 | 0.78 | 0.88 | 1.00 | 1.12 | 1.27 | 1.42 | 1.58 | | | | | |
| 9 | 0.87 | 0.99 | 1.10 | 1.25 | 1.39 | 1.55 | 1.72 | 1.90 | 2.10 | | | | |
| 10 | 1.11 | 1.24 | 1.39 | 1.54 | 1.71 | 1.89 | 2.07 | 2.28 | 2.50 | 2.74 | | | |
| 11 | 1.37 | 1.53 | 1.69 | 1.86 | 2.05 | 2.25 | 2.47 | 2.70 | 2.94 | 3.20 | 3.48 | | |
| 12 | 1.67 | 1.85 | 2.04 | 2.24 | 2.45 | 2.67 | 2.91 | 3.18 | 3.45 | 3.72 | 4.02 | 4.34 | |
| 13 | 2.01 | 2.22 | 2.43 | 2.65 | 2.89 | 3.14 | 3.41 | 3.69 | 3.99 | 4.30 | 4.64 | 4.97 | 5.34 |

Table 4.3 - Execution times in sec. of subroutine RUDY on PDP 15/35, when the number of states $n_x$ and the number of measurements $n_y$ are varied ($n_v = 1$).

## RUDY2.

The program RUDY exploits the fact that certain matrices are symmetric in order to obtain diagnostics. To save computing time and storage the diagnostics can be eliminated and we can compute one half of the symmetric matrices. This is done in the algorithm RUDY2 shown in Appendix E. This algorithm is shorter than RUDY because there is no computation of diagnostics ERR. The code for RUDY2 compiles into 684 cells. Subroutine MADD is not used by RUDY2, which means that RUDY2 is comparable to RUDY plus MADD. If all subprograms, Table 4.1, are included the algorithm requires 2956 cells. The number of cells required to store the data is shown in Table 4.2. The program also requires 1024 cells for the common block SLASK. Examples of execution times are given in Table 4.4.

RUDY2 is coded so that the required data set is given by (3.14). It is possible to decrease the data set according to (3.15) by a slight modification of the code.

## RUDY1 and RUDY3.

The program for the case that individual measurements are rejected is organized in the same way as RUDY. These are two versions which differ in the way the symmetry of the matrices are exploited. The program where symmetry is exploited for diagnosis is called RUDY1. See Appendix F. The program where symmetry is exploited to save computations and memory is called RUDY3. See Appendix G. RUDY1 requires 889 memory locations while RUDY3 requires 826. The number of cells for RUDY1 includes MADD, while RUDY3 does not use MADD. The computing times for the different algorithms are shown in Table 4.5 and Table 4.6.

| $n_x$ ↓ \ $n_y$ → | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0.04 | 0.07 | | | | | | | | | | | |
| 3 | 0.06 | 0.10 | 0.14 | | | | | | | | | | |
| 4 | 0.10 | 0.15 | 0.20 | 0.25 | | | | | | | | | |
| 5 | 0.15 | 0.21 | 0.27 | 0.34 | 0.41 | | | | | | | | |
| 6 | 0.21 | 0.28 | 0.35 | 0.43 | 0.52 | 0.61 | | | | | | | |
| 7 | 0.30 | 0.38 | 0.46 | 0.55 | 0.66 | 0.77 | 0.88 | | | | | | |
| 8 | 0.40 | 0.49 | 0.59 | 0.70 | 0.82 | 0.94 | 1.08 | 1.23 | | | | | |
| 9 | 0.51 | 0.62 | 0.74 | 0.86 | 1.00 | 1.14 | 1.29 | 1.46 | 1.63 | | | | |
| 10 | 0.66 | 0.78 | 0.92 | 1.06 | 1.21 | 1.38 | 1.55 | 1.73 | 1.93 | 2.14 | | | |
| 11 | 0.81 | 0.96 | 1.12 | 1.28 | 1.45 | 1.63 | 1.83 | 2.03 | 2.25 | 2.48 | 2.72 | | |
| 12 | 1.00 | 1.17 | 1.34 | 1.53 | 1.73 | 1.92 | 2.14 | 2.38 | 2.61 | 2.86 | 3.13 | 3.41 | |
| 13 | 1.22 | 1.41 | 1.60 | 1.81 | 2.03 | 2.26 | 2.50 | 2.76 | 3.02 | 3.30 | 3.59 | 3.90 | 4.22 |

Table 4.4 - Execution times in sec. of subroutine RUDY2 on PDP15/35, when the number of states $n_x$ and the number of measurements $n_y$ are varied ($n_u$ = 1).

| $n_x \downarrow$ / $n_y \rightarrow$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0.05 | 0.08 | | | | | | | | | | | |
| 3 | 0.10 | 0.14 | 0.18 | | | | | | | | | | |
| 4 | 0.17 | 0.21 | 0.26 | 0.32 | | | | | | | | | |
| 5 | 0.26 | 0.31 | 0.37 | 0.45 | 0.53 | | | | | | | | |
| 6 | 0.36 | 0.43 | 0.50 | 0.59 | 0.69 | 0.79 | | | | | | | |
| 7 | 0.50 | 0.59 | 0.67 | 0.77 | 0.89 | 1.01 | 1.14 | | | | | | |
| 8 | 0.67 | 0.77 | 0.88 | 0.99 | 1.12 | 1.26 | 1.41 | 1.57 | | | | | |
| 9 | 0.86 | 0.98 | 1.10 | 1.24 | 1.38 | 1.54 | 1.71 | 1.89 | 2.09 | | | | |
| 10 | 1.10 | 1.23 | 1.38 | 1.53 | 1.69 | 1.87 | 2.06 | 2.27 | 2.49 | 2.73 | | | |
| 11 | 1.36 | 1.51 | 1.68 | 1.85 | 2.04 | 2.24 | 2.46 | 2.68 | 2.93 | 3.19 | 3.46 | | |
| 12 | 1.67 | 1.85 | 2.03 | 2.23 | 2.44 | 2.66 | 2.90 | 3.15 | 3.42 | 3.72 | 4.02 | 4.33 | |
| 13 | 2.01 | 2.21 | 2.42 | 2.64 | 2.88 | 3.13 | 3.39 | 3.68 | 3.98 | 4.29 | 4.61 | 4.96 | 5.32 |

Table 4.5 - Execution times in sec. of subroutine RUDY1 on PDP 15/35, when the number of states $n_x$ and the number of measurements $n_y$ are varied ($n_u$ = 1).

| $n_x$ \ $n_y \rightarrow$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 0.04 | 0.06 | | | | | | | | | | | |
| 3 | 0.06 | 0.10 | 0.14 | | | | | | | | | | |
| 4 | 0.10 | 0.15 | 0.19 | 0.24 | | | | | | | | | |
| 5 | 0.15 | 0.21 | 0.26 | 0.33 | 0.40 | | | | | | | | |
| 6 | 0.21 | 0.27 | 0.34 | 0.42 | 0.50 | 0.59 | | | | | | | |
| 7 | 0.29 | 0.37 | 0.45 | 0.54 | 0.63 | 0.74 | 0.86 | | | | | | |
| 8 | 0.39 | 0.48 | 0.57 | 0.68 | 0.79 | 0.91 | 1.05 | 1.19 | | | | | |
| 9 | 0.50 | 0.61 | 0.71 | 0.83 | 0.96 | 1.11 | 1.25 | 1.41 | 1.58 | | | | |
| 10 | 0.64 | 0.76 | 0.89 | 1.03 | 1.17 | 1.33 | 1.50 | 1.68 | 1.87 | 2.07 | | | |
| 11 | 0.79 | 0.93 | 1.08 | 1.24 | 1.40 | 1.58 | 1.77 | 1.96 | 2.18 | 2.40 | 2.63 | | |
| 12 | 0.98 | 1.14 | 1.31 | 1.48 | 1.67 | 1.87 | 2.08 | 2.30 | 2.54 | 2.78 | 3.03 | 3.31 | |
| 13 | 1.18 | 1.37 | 1.55 | 1.75 | 1.96 | 2.18 | 2.42 | 2.67 | 2.92 | 3.19 | 3.47 | 3.77 | 4.08 |

Table 4.6 - Execution times in sec. of subroutine RUDY3 on PDP 15/35, when the number of states $n_x$ and the number of measurements $n_y$ are varied ($n_u = 1$).

## 5. REFERENCES.

[1] Åström, K.J.: Introduction to Stochastic Control Theory, Academic Press, 1970.

[2] Hagander, P., Källström, C., Åström, K.J.: Real Time Computing I - Implementing Linear Filters, Report 7106(B), Division of Automatic Control, Lund Institute of Technology, 1971.

# APPENDIX A

```
      SUBROUTINE GLIRE1(A,BK,C,AL,X,U,Y,NX,NU,NY)
C
C     COMPUTES THE NEW STATE ESTIMATE XE(T+1) AND THE NEW CONTROL
C     U(T+1) FROM XE(T) AND Y(T) FOR THE GENERAL LINEAR REGULATOR
C             XE(T+1)=A*XE(T)+B*U(T)+K*(Y(T)-C*XE(T))
C             U(T+1)=-L*XE(T+1)
C
C     CAN ALSO BE USED AS A PURE KALMAN FILTER BY PUTTING NU=0
C
C     AUTHOR KJ ASTROM 1971-07-22
C
C     A- MATRIX OF ORDER NX*NX
C     BK-MATRIX OF ORDER NX*(NU+NY) CONCATENATION OF B AND K
C     C- MATRIX OF ORDER NY*NX
C     AL-GAIN MATRIX -L OF ORDER NU*NX. NOTICE -L.
C     X- STATE ESTIMATE XE(T) OF DIMENSION NX, RETURNED AS XE(T+1)
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS THE NEW CONTROL
C     Y- MEASUREMENT VECTOR OF ORDER NY
C     NX-NUMBER OF STATES (MAX 64 GLISY )
C     NU-NUMBER OF CONTROL VARIABLES, NU=0 IF PURE KALMAN FILTER
C     NY-NUMBER OF MEASUREMENTS (MAX(NU+NY)=64)
C
C     UE-DUMMY VECTOR OF DIMENSION NU+NY CONCATENATION OF U
C        AND Y-C*XE STORED IN DUM7 OF THE COMMON BLOCK /SLASK/.
C        THE LAST FIELD DUM8 OF /SLASK/ CONTAINS XE AND THE FIRST
C        768 CELLS ARE NOT USED.
C
C     SUBROUTINES REQUIRED
C             GLISY
C
      DIMENSION A(1,1),BK(1,1),C(1,1),AL(1,1),X(1),U(1),Y(1)
      DIMENSION UE(64),DUMY(384),DUM8(64)
C
      COMMON /SLASK/DUMY,UE,DUM8
C
```

## APPENDIX B

```
      SUBROUTINE GLISY1(A,B,C,X,U,Y,NX,NU,NY)
C
C     COMPUTES THE NEW STATE X(T+1) AND THE NEW OUTPUT Y(T+1) FROM
C     X(T) AND U(T) FOR THE SYSTEM
C             X(T+1)=A*X(T)+B*U(T)
C             Y(T+1)=C*X(T+1)
C
C     AUTHOR KJ ASTROM 1971-07-22
C
C     A- MATRIX OF ORDER NX*NX
C     B- MATRIX OF ORDER NX*NU
C     C- MATRIX OF ORDER NY*NX
C     X- STATE VECTOR X(T) OF DIMENSION NX, RETURNED AS X(T+1)
C     U- INPUT VECTOR U(T) OF DIMENSION NU
C     Y- OUTPUT VECTOR Y(T+1) OF DIMENSION NY
C     NX-NUMBER OF STATES (MAX 64 SEE SECOND DIMENSION STATEMENT)
C     NU-NUMBER OF INPUTS (NO MAX)
C     NY-NUMBER OF OUTPUTS (NO MAX). PUT NY=0 IF COMPUTATION OF Y(T)
C        SHOULD BE SKIPPED.
C
C     XO-DUMMY VECTOR OF DIMENSION NX CONTAINING THE
C        THE NEW STATE VECTOR X(T+1). STORED IN DUM8 OF
C        THE COMMON BLOCK /SLASK/. THE FIRST 896 CELLS OF /SLASK/
C        ARE NOT USED.
C
C     SUBROUTINES REQUIRED
C             NONE
C
      DIMENSION A(1,1),B(1,1),C(1,1),X(1),U(1),Y(1)
      DIMENSION XO(64),DUMY(448)
C
      COMMON /SLASK/ DUMY,XO
C
```

## APPENDIX C

Computing time for GLISY 1 on PDP 15/35

| $n_x$ | $n_u$ | $n_y$ | Execution time (m sec) |
|---|---|---|---|
| 4 | 3 | 2 | 10.1 |
| 4 | 3 | 0 | 8.2 |
| 5 | 1 | 1 | 10.3 |
| 10 | 1 | 1 | 28.6 |
| 20 | 1 | 1 | 90.9 |
| 40 | 1 | 1 | 315.7 |
| 10 | 5 | 1 | 35.5 |
| 10 | 1 | 5 | 36.7 |
| 10 | 5 | 5 | 43.5 |
| 40 | 3 | 2 | 336.4 |

Computing time for GLIRE 1 on PDP 15/35

| $n_x$ | $n_u$ | $n_y$ | Execution time (m sec) | Execution time when $n_u$=0 (m sec) |
|---|---|---|---|---|
| 4 | 3 | 2 | 14.9 | 10.1 |
| 5 | 1 | 1 | 13.0 | 10.9 |
| 10 | 1 | 1 | 33.2 | 29.4 |
| 20 | 1 | 1 | 98.3 | 91.0 |
| 40 | 1 | 1 | 330.4 | 317.1 |
| 10 | 5 | 1 | 47.8 | 29.1 |
| 10 | 1 | 5 | 49.1 | 45.3 |
| 10 | 5 | 5 | 64.4 | 45.6 |
| 40 | 3 | 2 | 372.9 | 331.5 |

Note that the average addition and multiplication time on PDP 15/35 is about 200 $\mu$s.

## APPENDIX D

```
      SUBROUTINE RUDY(A,B,C,AL,R1,R21,R2,P,X,U,Y,NX,NU,NY,EPS,TEST,
     1   IND,ERR)
C
C     STATE ESTIMATION USING KALMAN FILTER.
C     THE FACT THAT SOME MATRICES ARE SYMMETRIC IS NOT CONSIDERED.
C     IF THE MEASUREMENT VECTOR DIFFERS TOO MUCH FROM THE ESTIMATE,
C     THE MEASUREMENTS ARE REJECTED.
C     AUTHOR, KJ ASTROM/C KALLSTROM 1971-11-15.
C
C     A- SYSTEM MATRIX OF ORDER NX*NX.
C     B- SYSTEM MATRIX OF ORDER NX*NU.
C     C- SYSTEM MATRIX OF ORDER NY*NX.
C     AL- FEED BACK MATRIX -L OF ORDER NU*NX.
C     R1- STATE NOISE COVARIANCE OF ORDER NX*NX.
C     R21- STATE/MEASUREMENT COVARIANCE OF ORDER NY*NX.
C     R2- MEASUREMENT NOISE COVARIANCE OF ORDER NY*NY.
C     P- COVARIANCE OF ESTIMATION ERROR P(T) OF ORDER NX*NX,
C        RETURNED AS P(T+1).
C     X- STATE ESTIMATE X(T) OF DIMENSION NX, RETURNED AS STATE
C        ESTIMATE X(T+1).
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS
C        THE NEW CONTROL.
C     Y- MEASUREMENT VECTOR OF DIMENSION NY, RETURNED CONTAINING
C        THE RESIDUALS Y-C*X(T).
C     NX- NUMBER OF STATES (MAX 13, MIN 2).
C     NU- NUMBER OF CONTROL VARIABLES (MAX 13, MIN 0). PUT NU=0
C        IF PURE KALMAN FILTER.
C     NY- NUMBER OF MEASUREMENTS (MAX 13,MIN 1).
C     EPS- TEST QUANTITY TO DECIDE IF C*P*CT+R2 IS SINGULAR.
C     TEST- TEST QUANTITY TO DECIDE IF THE MEASUREMENT VECTOR SHOULD
C        BE REJECTED.
C     IND- IS RETURNED 1 IF THE MEASUREMENT VECTOR IS REJECTED
C                      2 IF C*P*CT+R2 IS SINGULAR
C                      0 OTHERWISE
C     ERR- SYMMETRY ERROR OF P(T+1).
C
C     THE LAST 1014 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C            MMULT
C            MADD
C            NORM
C            DESYM
C            SOLVS
C
      DIMENSION A(1,1),B(1,1),C(1,1),AL(1,1),R1(1,1),
     1   R21(1,1),R2(1,1),X(1),U(1),Y(1),P(1,1)
C
      COMMON/SLASK/IDUM(10),S1(13,13),S2(13,13),S3(13,13)
C
```

## APPENDIX E

```
      SUBROUTINE RUDY2(A,B,C,AL,R1,R21,R2,P,X,U,Y,NX,NU,NY,EPS,TEST,
     1   IND)
C
C     STATE ESTIMATION USING KALMAN FILTER.
C     THE FACT THAT SOME MATRICES ARE SYMMETRIC IS CONSIDERED.
C     IF THE MEASUREMENT VECTOR DIFFERS TOO MUCH FROM THE ESTIMATE,
C     THE MEASUREMENTS ARE REJECTED.
C     AUTHOR, KJ ASTROM/C.KALLSTROM 1971-11-15.
C
C     A- SYSTEM MATRIX OF ORDER NX*NX.
C     B- SYSTEM MATRIX OF ORDER NX*NU.
C     C- SYSTEM MATRIX OF ORDER NY*NX.
C     AL- FEED BACK MATRIX -L OF ORDER NU*NX.
C     R1- STATE NOISE COVARIANCE OF ORDER NX*NX.
C     R21- STATE/MEASUREMENT COVARIANCE OF ORDER NY*NX.
C     R2- MEASUREMENT NOISE COVARIANCE OF ORDER NY*NY.
C     P- COVARIANCE OF ESTIMATION ERROR P(T) OF ORDER NX*NX,
C        RETURNED AS P(T+1).
C     X- STATE ESTIMATE X(T) OF DIMENSION NX, RETURNED AS STATE
C        ESTIMATE X(T+1).
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS
C        THE NEW CONTROL.
C     Y- MEASUREMENT VECTOR OF DIMENSION NY, RETURNED CONTAINING
C        THE RESIDUALS Y-C*X(T).
C     NX- NUMBER OF STATES (MAX 13, MIN 1).
C     NU- NUMBER OF CONTROL VARIABLES (MAX 13, MIN 0). PUT NU=0
C        IF PURE KALMAN FILTER.
C     NY- NUMBER OF MEASUREMENTS (MAX 13,MIN 1).
C     EPS- TEST QUANTITY TO DECIDE IF C*P*CT+R2 IS SINGULAR.
C     TEST- TEST QUANTITY TO DECIDE IF THE MEASUREMENT VECTOR SHOULD
C        BE REJECTED.
C     IND- IS RETURNED 1 IF THE MEASUREMENT VECTOR IS REJECTED
C                      2 IF C*P*CT+R2 IS SINGULAR
C                      0 OTHERWISE
C
C     THE LAST 1014 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C            MMULT
C            NORM
C            DESYM
C            SOLVS
C
      DIMENSION A(1,1),B(1,1),C(1,1),AL(1,1),R1(1,1),
     1   R21(1,1),R2(1,1),X(1),U(1),Y(1),P(1,1)
C
      COMMON/SLASK/IDUM(10),S1(13,13),S2(13,13),S3(13,13)
C
```

## APPENDIX F

```
      SUBROUTINE RUDY1(A,B,C,AL,R1,R21,R2,P,X,U,Y,NX,NU,NY,EPS,
      1                 TEST,IND,ERR,IY)
C
C     STATE ESTIMATION USING KALMAN FILTER.
C     THE FACT THAT SOME MATRICES ARE SYMMETRIC IS NOT CONSIDERED.
C     IF ANY MEASUREMENTS DIFFER TOO MUCH FROM ESTIMATION
C     THEY ARE REJECTED.
C     AUTHOR, KJ ASTROM/C.KALLSTROM 1971-11-15.
C
C     A- SYSTEM MATRIX OF ORDER NX*NX.
C     B- SYSTEM MATRIX OF ORDER NX*NU.
C     C- SYSTEM MATRIX OF ORDER NY*NX.
C     AL- FEED BACK MATRIX -L OF ORDER NU*NX.
C     R1- STATE NOISE COVARIANCE OF ORDER NX*NX.
C     R21- STATE/MEASUREMENT COVARIANCE OF ORDER NY*NX.
C     R2- MEASUREMENT NOISE COVARIANCE OF ORDER NY*NY.
C     P- COVARIANCE OF ESTIMATION ERROR P(T) OF ORDER NX*NX,
C        RETURNED AS P(T+1).
C     X- STATE ESTIMATE X(T) OF DIMENSION NX, RETURNED AS STATE
C        ESTIMATE X(T+1).
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS
C        THE NEW CONTROL.
C     Y- MEASUREMENT VECTOR OF DIMENSION NY.
C     NX- NUMBER OF STATES (MAX 13, MIN 1).
C     NU- NUMBER OF CONTROL VARIABLES (MAX 13, MIN 0). PUT NU=0
C        IF PURE KALMAN FILTER.
C     NY- NUMBER OF MEASUREMENTS (MAX 13,MIN 1).
C     EPS- TEST QUANTITY TO DECIDE IF C*P*CT+R2 IS SINGULAR.
C     TEST- TEST QUANTITY TO DECIDE IF ANY MEASUREMENTS SHOULD
C        BE REJECTED.
C     IND- RETURNED 1 IF ALL MEASUREMENTS ARE REJECTED
C                   2 IF C*P*CT+R2 IS SINGULAR
C                   0 OTHERWISE
C     ERR- SYMMETRY ERROR OF P(T+1).
C     IY- VECTOR OF DIMENSION NY CONTAINING  THE SUBSCRIPTS
C        OF THE ACCEPTED MEASUREMENTS.
C
C     THE LAST 1014 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C            MMULT
C            MADD
C            NORM
C            DESYM
C            SOLVS
C
      DIMENSION A(1,1), B(1,1),C(1,1),AL(1,1),R1(1,1),R21(1,1),
      1          R2(1,1),X(1),U(1),Y(1),P(1,1),IY(1)
C
      COMMON/SLASK/IDUM(10),S1(13,13),S2(13,13),S3(13,13)
C
```

## APPENDIX G

```
      SUBROUTINE RUDY3(A,B,C,AL,R1,R21,R2,P,X,U,Y,NX,NU,NY,EPS,
     1                 TEST,IND,IY)
C
C     STATE ESTIMATION USING KALMAN FILTER.
C     THE FACT THAT SOME MATRICES ARE SYMMETRIC IS CONSIDERED.
C     IF ANY MEASUREMENTS DIFFER TOO MUCH FROM ESTIMATION
C     THEY ARE REJECTED.
C     AUTHOR, KJ ASTROM/C. KALLSTROM 1971-11-15.
C
C     A- SYSTEM MATRIX OF ORDER NX*NX.
C     B- SYSTEM MATRIX OF ORDER NX*NU.
C     C- SYSTEM MATRIX OF ORDER NY*NX.
C     AL- FEED BACK MATRIX -L OF ORDER NU*NX.
C     R1- STATE NOISE COVARIANCE OF ORDER NX*NX.
C     R21- STATE/MEASUREMENT COVARIANCE OF ORDER NY*NX.
C     R2- MEASUREMENT NOISE COVARIANCE OF ORDER NY*NY.
C     P- COVARIANCE OF ESTIMATION ERROR P(T) OF ORDER NX*NX,
C        RETURNED AS P(T+1).
C     X- STATE ESTIMATE X(T) OF DIMENSION NX, RETURNED AS STATE
C        ESTIMATE X(T+1).
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS
C        THE NEW CONTROL.
C     Y- MEASUREMENT VECTOR OF DIMENSION NY.
C     NX- NUMBER OF STATES (MAX 13, MIN 1).
C     NU- NUMBER OF CONTROL VARIABLES (MAX 13, MIN 0). PUT NU=0
C        IF PURE KALMAN FILTER.
C     NY- NUMBER OF MEASUREMENTS (MAX 13,MIN 1).
C     EPS- TEST QUANTITY TO DECIDE IF C*P*CT+R2 IS SINGULAR.
C     TEST- TEST QUANTITY TO DECIDE IF ANY MEASUREMENTS SHOULD
C        BE REJECTED.
C     IND- RETURNED 1 IF ALL MEASUREMENTS ARE REJECTED
C                   2 IF C*P*CT+R2 IS SINGULAR
C                   0 OTHERWISE
C     IY- VECTOR OF DIMENSION NY CONTAINING  THE SUBSCRIPTS
C        OF THE ACCEPTED MEASUREMENTS.
C
C     THE LAST 1014 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C            MMULT
C            NORM
C            DESYM
C            SOLVS
C
      DIMENSION A(1,1), B(1,1),C(1,1),AL(1,1),R1(1,1),R21(1,1),
     1          R2(1,1),X(1),U(1),Y(1),P(1,1),IY(1)
C
      COMMON/SLASK/IDUM(10),S1(13,13),S2(13,13),S3(13,13)
C
```

```
      SUBROUTINE RUDY(A,B,C,AL,R1,R21,R2,P,X,U,Y,NX,NU,NY,EPS,TEST,
     1   IND,ERR)
C
C     STATE ESTIMATION USING KALMAN FILTER.
C     THE FACT THAT SOME MATRICES ARE SYMMETRIC IS NOT CONSIDERED.
C     IF THE MEASUREMENT VECTOR DIFFERS TOO MUCH FROM THE ESTIMATE,
C     THE MEASUREMENTS ARE REJECTED.
C     AUTHOR, KJ ASTROM/C KALLSTROM 1971-11-15.
C
C     A- SYSTEM MATRIX OF ORDER NX*NX.
C     B- SYSTEM MATRIX OF ORDER NX*NU.
C     C- SYSTEM MATRIX OF ORDER NY*NX.
C     AL- FEED BACK MATRIX -L OF ORDER NU*NX.
C     R1- STATE NOISE COVARIANCE OF ORDER NX*NX.
C     R21- STATE/MEASUREMENT COVARIANCE OF ORDER NY*NX.
C     R2- MEASUREMENT NOISE COVARIANCE OF ORDER NY*NY.
C     P- COVARIANCE OF ESTIMATION ERROR P(T) OF ORDER NX*NX,
C        RETURNED AS P(T+1).
C     X- STATE ESTIMATE X(T) OF DIMENSION NX, RETURNED AS STATE
C        ESTIMATE X(T+1).
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS
C        THE NEW CONTROL.
C     Y- MEASUREMENT VECTOR OF DIMENSION NY, RETURNED CONTAINING
C        THE RESIDUALS Y-C*X(T).
C     NX- NUMBER OF STATES (MAX 13, MIN 2).
C     NU- NUMBER OF CONTROL VARIABLES (MAX 13, MIN 0). PUT NU=0
C        IF PURE KALMAN FILTER.
C     NY- NUMBER OF MEASUREMENTS (MAX 13,MIN 1).
C     EPS- TEST QUANTITY TO DECIDE IF C*P*CT+R2 IS SINGULAR.
C     TEST- TEST QUANTITY TO DECIDE IF THE MEASUREMENT VECTOR SHOULD
C        BE REJECTED.
C     IND- IS RETURNED 1 IF THE MEASUREMENT VECTOR IS REJECTED
C                      2 IF C*P*CT+R2 IS SINGULAR
C                      0 OTHERWISE
C     ERR- SYMMETRY ERROR OF P(T+1).
C
C     THE LAST 1014 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C              MMULT
C              MADD
C              NORM
C              DESYM
C              SOLVS
C
      DIMENSION A(1,1),B(1,1),C(1,1),AL(1,1),R1(1,1),
     1   R21(1,1),R2(1,1),X(1),U(1),Y(1),P(1,1)
C
      COMMON/SLASK/IDUM(10),S1(13,13),S2(13,13),S3(13,13)
C
      IS=13
      R=-1.
      IND=0
C
C     EVALUATE RESIDUALS Y-C*X(T)
C
      DO 2 I=1,NY
    2 Y(I)=Y(I)-SCAPRO(C(I,1),NY,X(1),1,NX)          ✓
      CALL MOVE(Y(1),S3(1,1),NY+NY)
C
C     Y AND THE FIRST COLUMN OF S3 NOW CONTAINS THE RESIDUALS
C     Y-C*X(T)
C
```

```
C         EVALUATE MEASUREMENT COVARIANCE C*P*CT+R2
C
          CALL MMULT(P,C,S2,NX,NX,NY,NX,NY,0,1)
          CALL MMULT(C,S2,S1,NY,NX,NY,NY,IS,0,0)
          CALL MADD(S1,R2,S1,NY,NY,0)
C
C         S1 NOW CONTAINS C*P*CT+R2
C
          CALL NORM(S1,NY,IS,ANORM)
          EP=ANORM*EPS
C
          CALL DESYM(S1,S2,NY,EP,IRANK,IS)
C
C         S2 NOW CONTAINS THE TRIANGULAR DECOMPOSITION OF C*P*CT+R2
C
          IF(IRANK-NY) 4,6,4
4         IND=1
          GO TO 8
C
C         TEST IF MEASUREMENTS CLOSE TO ESTIMATES
C
6         CALL SOLVS(S2,S3,S1,NY,1,IS)
          R=TEST-SCAPRO(S3(1,1),1,S1(1,1),1,NY)
C
8         CALL MMULT(A,P,S1,NX,NX,NX,NX,NX,0,0)
C
C         S1 NOW CONTAINS A*P
C
          IF(R) 10,20,20
10        IND=IND+1
          GO TO 30
C
C         COMPUTE FILTER GAIN KT=(C*P*CT+R2)(-1)*(C*PT*AT+R21)
C
20        CALL MMULT(C,S1,P,NY,NX,NX,NY,IS,0,1)
C
          CALL MADD(R21,P,P,NY,NX,0)
C
C         P NOW CONTAINS C*PT*AT+R21
C
          CALL SOLVS(S2,P,S3,NY,NX,IS)
C
C         S3 NOW CONTAINS KT
C
C         UPDATE STATE X=A*X+B*U+K*(Y-C*X) OR X=A*X+B*U
C
30        DO 38 I=1,NX
          R=SCAPRO(A(I,1),NX,X(1),1,NX)
          IF(NU) 34,34,32
32        R=R+SCAPRO(B(I,1),NX,U(1),1,NU)
34        IF(IND) 36,36,38
36        R=R+SCAPRO(S3(1,I),1,Y(1),1,NY)
38        S2(I,1)=R
C
          CALL MOVE(S2(1,1),X(1),NX+NX)
C
          IF(NU) 44,44,40
C
C         COMPUTE NEW CONTROL U=AL*X
C
40        DO 42 I=1,NU
42        U(I)=SCAPRO(AL(I,1),NU,X(1),1,NX)
C
44        IF(IND) 50,50,60
```

```
C
C       UPDATE COVARIANCE P=A*P*(A-K*C)T+R1-R12*KT IF IND=0
C
50      CALL MMULT(R21,S3,P,NX,NY,NX,NY,IS,1,0)
C
C       P NOW CONTAINS R12*KT
C
        CALL MMULT(S3,C,S2,NX,NY,NX,IS,NY,1,0)
        CALL MADD(A,S2,S2,NX,NX,1)
C
C       S2 NOW CONTAINS A-K*C
C
        CALL MMULT(S1,S2,S3,NX,NX,NX,IS,IS,0,1)
        CALL MADD(S3,P,P,NX,NX,1)
C
C       P NOW CONTAINS A*P*(A-K*C)T-R12*KT
C
        GO TO 62
C
C       UPDATE COVARIANCE P=A*P*AT+R1 IF IND=1 OR IND=2
C
60      CALL MMULT(S1,A,P,NX,NX,NX,IS,NX,0,1)
C
62      CALL MADD(P,R1,P,NX,NX,0)
C
C       P NOW CONTAINS A*P*(A-K*C)T+R1-R12*KT OR A*P*AT+R1
C
C       COMPUTE SYMMETRY ERROR OF P(T+1) DEFINED AS
C       ERR = NORM(P-PT)/NORM(P+PT) AND SYMMETRIZE P(T+1)
C
        K=NX-1
        DO 70 I=1,K
        II=I+1
        DO 70 J=II,NX
        R=P(I,J)
        RR=P(J,I)
        Q1=(R+RR)/2.
        Q2=R-RR
        P(I,J)=Q1
        P(J,I)=Q1
        S1(I,J)=Q2
70      S1(J,I)=-Q2
C
        DO 72 I=1,NX
72      S1(I,I)=0.
C
        CALL NORM(S1,NX,IS,Q1)
C
        CALL NORM(P,NX,NX,Q2)
C
        ERR=Q1/(Q2+Q2)
C
        RETURN
        END
```

```
      SUBROUTINE RUDY1(A,B,C,AL,R1,R21,R2,P,X,U,Y,NX,NU,NY,EPS,
     1                 TEST,IND,ERR,IY)
C
C     STATE ESTIMATION USING KALMAN FILTER.
C     THE FACT THAT SOME MATRICES ARE SYMMETRIC IS NOT CONSIDERED.
C     IF ANY MEASUREMENTS DIFFER TOO MUCH FROM ESTIMATION
C     THEY ARE REJECTED.
C     AUTHOR, KJ ASTROM/C.KALLSTROM 1971-11-15.
C
C     A- SYSTEM MATRIX OF ORDER NX*NX.
C     B- SYSTEM MATRIX OF ORDER NX*NU.
C     C- SYSTEM MATRIX OF ORDER NY*NX.
C     AL- FEED BACK MATRIX -L OF ORDER NU*NX.
C     R1- STATE NOISE COVARIANCE OF ORDER NX*NX.
C     R21- STATE/MEASUREMENT COVARIANCE OF ORDER NY*NX.
C     R2- MEASUREMENT NOISE COVARIANCE OF ORDER NY*NY.
C     P- COVARIANCE OF ESTIMATION ERROR P(T) OF ORDER NX*NX,
C        RETURNED AS P(T+1).
C     X- STATE ESTIMATE X(T) OF DIMENSION NX, RETURNED AS STATE
C        ESTIMATE X(T+1).
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS
C        THE NEW CONTROL.
C     Y- MEASUREMENT VECTOR OF DIMENSION NY.
C     NX- NUMBER OF STATES (MAX 13, MIN 1).
C     NU- NUMBER OF CONTROL VARIABLES (MAX 13, MIN 0). PUT NU=0
C        IF PURE KALMAN FILTER.
C     NY- NUMBER OF MEASUREMENTS (MAX 13,MIN 1).
C     EPS- TEST QUANTITY TO DECIDE IF C*P*CT+R2 IS SINGULAR.
C     TEST- TEST QUANTITY TO DECIDE IF ANY MEASUREMENTS SHOULD
C        BE REJECTED.
C     IND- RETURNED 1 IF ALL MEASUREMENTS ARE REJECTED
C                   2 IF C*P*CT+R2 IS SINGULAR
C                   0 OTHERWISE
C     ERR- SYMMETRY ERROR OF P(T+1).
C     IY- VECTOR OF DIMENSION NY CONTAINING  THE SUBSCRIPTS
C        OF THE ACCEPTED MEASUREMENTS.
C
C     THE LAST 1014 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C            MMULT
C            MADD
C            NORM
C            DESYM
C            SOLVS
C
      DIMENSION A(1,1), B(1,1),C(1,1),AL(1,1),R1(1,1),R21(1,1),
     1          R2(1,1),X(1),U(1),Y(1),P(1,1),IY(1)
C
      COMMON/SLASK/IDUM(10),S1(13,13),S2(13,13),S3(13,13)
C
      IS=13
      IND=0
C
C     INITIALIZE IY AND EVALUATE RESIDUALS Y-C*X(T)
C
      DO 2 I=1,NY
      IY(I)=0
    2 Y(I)=Y(I)-SCAPRO(C(I,1),NY,X(1),1,NX)
C
C     Y NOW CONTAINS THE RESIDUALS Y-C*X(T)
C
C     EVALUATE MEASUREMENT COVARIANCE C*P*CT+R2
```

```
C
      CALL MMULT(P,C,S1,NX,NX,NY,NX,NY,0,1)
      CALL MMULT(C,S1,S2,NY,NX,NY,NY,IS,0,0)
      CALL MADD(R2,S2,S2,NY,NY,0)
C
C     S2 NOW CONTAINS C*P*CT+R2
C
C     TEST WHAT MEASUREMENTS SHOULD BE ACCEPTED
C
      I1=0
      DO 6 I=1,NY
      R=Y(I)*Y(I)/S2(I,I)
      IF(R-TEST)4,4,6
 4    I1=I1+1
      IY(I1)=I
      Y(I1)=Y(I)
 6    CONTINUE
      IF(I1)14,14,8
 8    NY1=I1
C
C     THE VECTOR IY CONTAINS THE INDICES OF THE ACCEPTED MEASUREMENTS,
C     NY1 IS THE NUMBER OF ACCEPTED MEASUREMENTS AND THE NY1 FIRST
C     COMPONENTS OF Y CONTAINS RESIDUALS OF ACCEPTED MEASUREMENTS
C
C     COLLAPSE S2
C
      DO 10 I=1,NY1
      II=IY(I)
      DO 10 J=1,NY1
      JJ=IY(J)
 10   S3(I,J)=S2(II,JJ)
C
C     S3 NOW CONTAINS COLLAPSED C*P*CT+R2, I.E. THOSE ELEMENTS THAT
C     CORRESPOND TO ACCEPTED MEASUREMENTS
C
      CALL NORM(S3,NY1,IS,ANORM)
      EP=ANORM*EPS
      CALL DESYM(S3,S2,NY1,EP, IRANK,IS)
C
C     S2 NOW CONTAINS THE TRIANGULAR DECOMPOSITION OF
C     COLLAPSED C*P*CT+R2
C
      IF(IRANK-NY1)12,16,12
 12   IND=2
      GO TO 16
 14   IND=1
 16   CALL MMULT(A,P,S1,NX,NX,NX,NX,NX,0,0)
C
C     S1 NOW CONTAINS A*P
C
      IF(IND)20,20,30
C
C     COMPUTE COLLAPSED FILTER GAIN
C
 20   DO 22 I=1,NY1
      II=IY(I)
      DO 22 J=1,NX
 22   P(I,J)=SCAPRO(C(II,1),NY,S1(J,1),IS,NX)+R21(II,J)
      CALL SOLVS(S2,P,S3,NY1,NX,IS)
C
C     S3 NOW CONTAINS COLLAPSED KT
C
C     UPDATE STATE
C
```

```
 30     DO 38 I=1,NX
        R=SCAPRO(A(I,1),NX,X(1),1,NX)
        IF(NU)34,34,32
 32     R=R+SCAPRO(B(I,1),NX,U(1),1,NU)
 34     IF(IND)36,36,38
 36     R=R+SCAPRO(S3(1,I),1,Y(1),1,NY1)
 38     S2(I,1)=R
        CALL MOVE(S2(1,1),X(1),NX+NX)
C
        IF(NU) 50,50,40
C
C       COMPUTE NEW CONTROL U=AL*X
C
 40     DO 42 I=1,NU
 42     U(I)=SCAPRO(AL(I,1),NU,X(1),1,NX)
C
 50     IF(IND)52,52,60
C
 52     S2(1,1)=0.
        CALL MOVE(S2(1,1),S2(2,1),2*IS*IS-2)
C
        DO 54 I=1,NY1
        II=IY(I)
        DO 54 J=1,NX
 54     S2(II,J)=S3(I,J)
C
C       S2 NOW CONTAINS KT OF FULL RANK
C
C       UPDATE COVARIANCE P=A*P*(A-K*C)T+R1-R12*KT IF IND=0
C
        CALL MMULT(R21,S2,P,NX,NY,NX,NY,IS,1,0)
C
C       P NOW CONTAINS R12*KT
C
        CALL MMULT(S2,C,S3,NX,NY,NX,IS,NY,1,0)
        CALL MADD(A,S3,S2,NX,NX,1)
C
C       S2 NOW CONTAINS A-K*C
C
        CALL MMULT(S1,S2,S3,NX,NX,NX,IS,IS,0,1)
        CALL MADD(S3,P,P,NX,NX,1)
C
C       P NOW CONTAINS A*P*(A-K*C)T-R12*KT
C
        GO TO 62
C
C       UPDATE COVARIANCE P=A*P*AT+R1 IF IND=1 OR IND=2
C
 60     CALL MMULT(S1,A,P,NX,NX,NX,IS,NX,0,1)
C
 62     CALL MADD(P,R1,P,NX,NX,0)
C
C       P NOW CONTAINS A*P*(A-K*C)T + R1 - R12*KT OR
C       A*P*AT + R1
C
C       COMPUTE SYMMETRY ERROR OF P(T+1) DEFINED AS
C       ERR = NORM(P-PT)/NORM(P+PT) AND SYMMETRIZE P(T+1)
C
        K=NX-1
        DO 70 I=1,K
        II=I+1
        DO 70 J=II,NX
        R=P(I,J)
        RR=P(J,I)
```

```
      Q1=(R+RR)/2.
      Q2=R-RR
      P(I,J)=Q1
      P(J,I)=Q1
      S1(I,J)=Q2
70    S1(J,I)=-Q2
C
      DO 72 I=1,NX
72    S1(I,I)=0.
C
      CALL NORM(S1,NX,IS,Q1)
C
      CALL NORM(P,NX,NX,Q2)
C
      ERR=Q1/(Q2+Q2)
C
      RETURN
      END
      SUBROUTINE RUDY2(A,B,C,AL,R1,R21,R2,P,X,U,Y,NX,NU,NY,EPS,TEST,
     1   IND)
C
C     STATE ESTIMATION USING KALMAN FILTER.
C     THE FACT THAT SOME MATRICES ARE SYMMETRIC IS CONSIDERED.
C     IF THE MEASUREMENT VECTOR DIFFERS TOO MUCH FROM THE ESTIMATE,
C     THE MEASUREMENTS ARE REJECTED.
C     AUTHOR, KJ ASTROM/C.KALLSTROM 1971-11-15.
C
C     A- SYSTEM MATRIX OF ORDER NX*NX.
C     B- SYSTEM MATRIX OF ORDER NX*NU.
C     C- SYSTEM MATRIX OF ORDER NY*NX.
C     AL- FEED BACK MATRIX -L OF ORDER NU*NX.
C     R1- STATE NOISE COVARIANCE OF ORDER NX*NX.
C     R21- STATE/MEASUREMENT COVARIANCE OF ORDER NY*NX.
C     R2- MEASUREMENT NOISE COVARIANCE OF ORDER NY*NY.
C     P- COVARIANCE OF ESTIMATION ERROR P(T) OF ORDER NX*NX,
C        RETURNED AS P(T+1).
C     X- STATE ESTIMATE X(T) OF DIMENSION NX, RETURNED AS STATE
C        ESTIMATE X(T+1).
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS
C        THE NEW CONTROL.
C     Y- MEASUREMENT VECTOR OF DIMENSION NY, RETURNED CONTAINING
C        THE RESIDUALS Y-C*X(T).
C     NX- NUMBER OF STATES (MAX 13, MIN 1).
C     NU- NUMBER OF CONTROL VARIABLES (MAX 13, MIN 0). PUT NU=0
C        IF PURE KALMAN FILTER.
C     NY- NUMBER OF MEASUREMENTS (MAX 13,MIN 1).
C     EPS- TEST QUANTITY TO DECIDE IF C*P*CT+R2 IS SINGULAR.
C     TEST- TEST QUANTITY TO DECIDE IF THE MEASUREMENT VECTOR SHOULD
C        BE REJECTED.
C     IND- IS RETURNED 1 IF THE MEASUREMENT VECTOR IS REJECTED
C                      2 IF C*P*CT+R2 IS SINGULAR
C                      0 OTHERWISE
C
C     THE LAST 1014 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C              MMULT
C              NORM
C              DESYM
C              SOLVS
C
      DIMENSION A(1,1),B(1,1),C(1,1),AL(1,1),R1(1,1),
     1   R21(1,1),R2(1,1),X(1),U(1),Y(1),P(1,1)
C
      COMMON/SLASK/IDUM(10),S1(13,13),S2(13,13),S3(13,13)
C
      IS=13
      R=-1.
      IND=0
C
C     EVALUATE RESIDUALS Y-C*X(T)
C
      DO 2 I=1,NY
    2 Y(I)=Y(I)-SCAPRO(C(I,1),NY,X(1),1,NX)
      CALL MOVE(Y(1),S3(1,1),NY+NY)
C
C     Y AND THE FIRST COLUMN OF S3 NOW CONTAINS RESIDUALS Y-C*X(T)
C
C     EVALUATE MEASUREMENT COVARIANCE C*P*CT+R2
C
      CALL MMULT(P,C,S2,NX,NX,NY,NX,NY,0,1)
      DO 4 I=1,NY
      DO 4 J=I,NY
      RR=R2(I,J)+SCAPRO(C(I,1),NY,S2(1,J),1,NX)
      S1(I,J)=RR
4     S1(J,I)=RR
C
C     S1 NOW CONTAINS C*P*CT+R2
C
      CALL NORM(S1,NY,IS,ANORM)
      EP=ANORM*EPS
C
      CALL DESYM(S1,S2,NY,EP,IRANK,IS)
C
C     S2 NOW CONTAINS THE TRIANGULAR DECOMPOSITION OF C*P*CT+R2
C
      IF(IRANK-NY) 6,8,6
6     IND=1
      GO TO 10
C
C     TEST IF MEASUREMENTS CLOSE TO ESTIMATES
C
8     CALL SOLVS(S2,S3,S1,NY,1,IS)
      R=TEST-SCAPRO(S3(1,1),1,S1(1,1),1,NY)
C
10    CALL MMULT(A,P,S1,NX,NX,NX,NX,NX,0,0)
C
C     S1 NOW CONTAINS A*P
C
      IF(R) 12,20,20
12    IND=IND+1
      GO TO 30
C
C     COMPUTE FILTER GAIN KT=(C*P*CT+R2)(-1)*(C*PT*AT+R21)
C
20    CALL MMULT(C,S1,P,NY,NX,NX,NY,IS,0,1)
C
      DO 22 I=1,NY
      DO 22 J=1,NX
22    P(I,J)=R21(I,J)+P(I,J)
C
C     P NOW CONTAINS C*PT*AT+R21
C
      CALL SOLVS(S2,P,S3,NY,NX,IS)
C
C     S3 NOW CONTAINS KT
C
C     UPDATE STATE X=A*X+B*U+K*(Y-C*X) OR X=A*X+B*U
C
30    DO 38 I=1,NX
      R=SCAPRO(A(I,1),NX,X(1),1,NX)
      IF(NU) 34,34,32
32    R=R+SCAPRO(B(I,1),NX,U(1),1,NU)
34    IF(IND) 36,36,38
36    R=R+SCAPRO(S3(1,I),1,Y(1),1,NY)
38    S2(I,1)=R
C
      CALL MOVE(S2(1,1),X(1),NX+NX)
C
      IF(NU) 44,44,40
C
C     COMPUTE NEW CONTROL U=AL*X
C
40    DO 42 I=1,NU
42    U(I)=SCAPRO(AL(I,1),NU,X(1),1,NX)
```

```
C
44      IF(IND) 50,50,60
C
C       UPDATE COVARIANCE P=A*P*(A-K*C)T+R1-R12*KT  IF  IND=0
C
50      CALL MMULT(S3,C,S2,NX,NY,NX,IS,NY,1,0)
        DO 52 I=1,NX
        DO 52 J=1,NX
52      S2(I,J)=A(I,J)-S2(I,J)
C
C       S2 NOW CONTAINS A-K*C
C
        DO 54 I=1,NX
        DO 54 J=I,NX
        RR=R1(I,J)+SCAPRO(S1(I,1),IS,S2(J,1),IS,NX)-
       1SCAPRO(R21(1,I),1,S3(1,J),1,NY)
        P(I,J)=RR
54      P(J,I)=RR
C
C       P NOW CONTAINS A*P*(A-K*C)T+R1-R12*KT
C
        GO TO 99
C
C       UPDATE COVARIANCE P=A*P*AT+R1  IF  IND=1 OR IND=2
C
60      DO 62 I=1,NX
        DO 62 J=I,NX
        RR=SCAPRO(S1(I,1),IS,A(J,1),NX,NX)+R1(I,J)
        P(I,J)=RR
62      P(J,I)=RR
C
C       P NOW CONTAINS A*P*AT+R1
C
99      RETURN
        END
```

```
      SUBROUTINE RUDY3(A,B,C,AL,R1,R21,R2,P,X,U,Y,NX,NU,NY,EPS,
     1                TEST,IND,IY)
C
C     STATE ESTIMATION USING KALMAN FILTER.
C     THE FACT THAT SOME MATRICES ARE SYMMETRIC IS CONSIDERED.
C     IF ANY MEASUREMENTS DIFFER TOO MUCH FROM ESTIMATION
C     THEY ARE REJECTED.
C     AUTHOR, KJ ASTROM/C. KALLSTROM 1971-11-15.
C
C     A- SYSTEM MATRIX OF ORDER NX*NX.
C     B- SYSTEM MATRIX OF ORDER NX*NU.
C     C- SYSTEM MATRIX OF ORDER NY*NX.
C     AL- FEED BACK MATRIX -L OF ORDER NU*NX.
C     R1- STATE NOISE COVARIANCE OF ORDER NX*NX.
C     R21- STATE/MEASUREMENT COVARIANCE OF ORDER NY*NX.
C     R2- MEASUREMENT NOISE COVARIANCE OF ORDER NY*NY.
C     P- COVARIANCE OF ESTIMATION ERROR P(T) OF ORDER NX*NX,
C        RETURNED AS P(T+1).
C     X- STATE ESTIMATE X(T) OF DIMENSION NX, RETURNED AS STATE
C        ESTIMATE X(T+1).
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS
C        THE NEW CONTROL.
C     Y- MEASUREMENT VECTOR OF DIMENSION NY.
C     NX- NUMBER OF STATES (MAX 13, MIN 1).
C     NU- NUMBER OF CONTROL VARIABLES (MAX 13, MIN 0). PUT NU=0
C        IF PURE KALMAN FILTER.
C     NY- NUMBER OF MEASUREMENTS (MAX 13,MIN 1).
C     EPS- TEST QUANTITY TO DECIDE IF C*P*CT+R2 IS SINGULAR.
C     TEST- TEST QUANTITY TO DECIDE IF ANY MEASUREMENTS SHOULD
C        BE REJECTED.
C     IND- RETURNED 1 IF ALL MEASUREMENTS ARE REJECTED
C                   2 IF C*P*CT+R2 IS SINGULAR
C                   0 OTHERWISE
C     IY- VECTOR OF DIMENSION NY CONTAINING  THE SUBSCRIPTS
C        OF THE ACCEPTED MEASUREMENTS.
C
C     THE LAST 1014 CELLS OF THE COMMON BLOCK /SLASK/ ARE USED.
C
C     SUBROUTINE REQUIRED
C            MMULT
C            NORM
C            DESYM
C            SOLVS
C
      DIMENSION A(1,1), B(1,1),C(1,1),AL(1,1),R1(1,1),R21(1,1),
     1          R2(1,1),X(1),U(1),Y(1),P(1,1),IY(1)
C
      COMMON/SLASK/IDUM(10),S1(13,13),S2(13,13),S3(13,13)
C
      IS=13
      IND=0
C
C     INITIALIZE IY AND EVALUATE RESIDUALS Y-C*X(T)
C
      DO 2 I=1,NY
      IY(I)=0
    2 Y(I)=Y(I)-SCAPRO(C(I,1),NY,X(1),1,NX)
C
C     Y NOW CONTAINS THE RESIDUALS Y-C*X(T)
C
C     EVALUATE MEASUREMENT COVARIANCE C*P*CT+R2
C
      CALL MMULT(P,C,S1,NX,NX,NY,NX,NY,0,1)
      DO 4 I=1,NY
      DO 4 J=I,NY
      RR=R2(I,J)+SCAPRO(C(I,1),NY,S1(1,J),1,NX)
      S2(I,J)=RR
4     S2(J,I)=RR
C
C     S2 NOW CONTAINS C*P*CT+R2
C
C     TEST WHAT MEASUREMENTS SHOULD BE ACCEPTED
C
      I1=0
      DO 8 I=1,NY
      R=Y(I)*Y(I)/S2(I,I)
      IF(R-TEST)6,6,8
 6    I1=I1+1
      IY(I1)=I
      Y(I1)=Y(I)
 8    CONTINUE
      IF(I1)16,16,10
 10   NY1=I1
C
C     THE VECTOR IY CONTAINS THE INDICES OF ACCEPTED MEASUREMENTS,
C     NY1 IS THE NUMBER OF ACCEPTED MEASUREMENTS AND THE NY1 FIRST
C     COMPONENTS OF Y CONTAINS RESIDUALS OF ACCEPTED MEASUREMENTS
C
C     COLLAPSE S2
C
      DO 12 I=1,NY1
      II=IY(I)
      DO 12 J=1,NY1
      JJ=IY(J)
 12   S3(I,J)=S2(II,JJ)
C
C     S3 NOW CONTAINS COLLAPSED C*P*CT+R2
C
      CALL NORM(S3,NY1,IS,ANORM)
      EP=ANORM*EPS
      CALL DESYM(S3,S2,NY1,EP, IRANK,IS)
C
C     S2 NOW CONTAINS THE TRIANGULAR DECOMPOSITION OF
C     COLLAPSED C*P*CT+R2
C
      IF(IRANK-NY1)14,18,14
 14   IND=2
      GO TO 18
 16   IND=1
 18   CALL MMULT(A,P,S1,NX,NX,NX,NX,NX,0,0)
C
C     S1 NOW CONTAINS A*P
C
      IF(IND)20,20,30
C
C     COMPUTE COLLAPSED FILTER GAIN
C
 20   DO 22 I=1,NY1
      II=IY(I)
      DO 22 J=1,NX
 22   P(I,J)=SCAPRO(C(II,1),NY,S1(J,1),IS,NX)+R21(II,J)
      CALL SOLVS(S2,P,S3,NY1,NX,IS)
C
C     S3 NOW CONTAINS COLLAPSED KT
C
C     UPDATE STATE
C
```

```
 30     DO 38 I=1,NX
        R=SCAPRO(A(I,1),NX,X(1),1,NX)
        IF(NU)34,34,32
 32     R=R+SCAPRO(B(I,1),NX,U(1),1,NU)
 34     IF(IND)36,36,38
 36     R=R+SCAPRO(S3(1,I),1,Y(1),1,NY1)
 38     S2(I,1)=R
        CALL MOVE(S2(1,1),X(1),NX+NX)
C
        IF(NU) 50,50,40
C
C       COMPUTE NEW CONTROL U=AL*X
C
 40     DO 42 I=1,NU
 42     U(I)=SCAPRO(AL(I,1),NU,X(1),1,NX)
C
 50     IF(IND)52,52,60
 52     S2(1,1)=0.
        CALL MOVE(S2(1,1),S2(2,1),2*IS*IS-2)
C
        DO 54 I=1,NY1
        II=IY(I)
        DO 54 J=1,NX
 54     S2(II,J)=S3(I,J)
C
C       S2 NOW CONTAINS KT OF FULL RANK
C
C       UPDATE COVARIANCE P=A*P*(A-K*C)T+R1-R12*KT IF IND=0
C
        CALL MMULT(S2,C,S3,NX,NY,NX,IS,NY,1,0)
C
        DO 56 I=1,NX
        DO 56 J=1,NX
56      S3(I,J)=A(I,J)-S3(I,J)
C
C       S3 NOW CONTAINS A-K*C
C
        DO 58 I=1,NX
        DO 58 J=I,NX
        RR=R1(I,J)+SCAPRO(S1(I,1),IS,S3(J,1),IS,NX)-
        1   SCAPRO(R21(1,I),1,S2(1,J),1,NY)
        P(I,J)=RR
58      P(J,I)=RR
C
C       P NOW CONTAINS A*P*(A-K*C)T + R1 - R12*KT
C
        GO TO 99
C
60      DO 62 I=1,NX
        DO 62 J=I,NX
        RR=SCAPRO(S1(I,1),IS,A(J,1),NX,NX)+R1(I,J)
        P(I,J)=RR
62      P(J,I)=RR
C
C       P NOW CONTAINS A*P*AT + R1
C
99      RETURN
        END
```

```
      SUBROUTINE MADD(A,B,C,N,M,MIN)
C
C     SUBROUTINE TO COMPUTE C=A+B OR C=A-B.
C     AUTHOR, C.KALLSTROM 1971-02-04.
C
C     A- MATRIX OF ORDER N*M.
C     B- MATRIX OF ORDER N*M.
C     C- MATRIX OF ORDER N*M.
C     N- NUMBER OF ROWS (NO MAX,MIN 1).
C     M- NUMBER OF COLUMNS (NO MAX,MIN 1).
C     MIN- PUT MIN=0 IF C=A+B AND MIN=1 IF C=A-B.
C
C     SUBROUTINE REQUIRED
C            NONE
C
      DIMENSION A(1,1),B(1,1),C(1,1)
C
      DO 10 I=1,N
      DO 10 J=1,M
      IF(MIN) 14,14,12
12    C(I,J)=A(I,J)-B(I,J)
      GO TO 10
14    C(I,J)=A(I,J)+B(I,J)
10    CONTINUE
C
      RETURN
      END
      SUBROUTINE GLISY1(A,B,C,X,U,Y,NX,NU,NY)
C
C     COMPUTES THE NEW STATE X(T+1) AND THE NEW OUTPUT Y(T+1) FROM
C     X(T) AND U(T) FOR THE SYSTEM
C             X(T+1)=A*X(T)+B*U(T)
C             Y(T+1)=C*X(T+1)
C
C     AUTHOR KJ ASTROM 1971-07-22
C
C     A- MATRIX OF ORDER NX*NX
C     B- MATRIX OF ORDER NX*NU
C     C- MATRIX OF ORDER NY*NX
C     X- STATE VECTOR X(T) OF DIMENSION NX, RETURNED AS X(T+1)
C     U- INPUT VECTOR U(T) OF DIMENSION NU
C     Y- OUTPUT VECTOR Y(T+1) OF DIMENSION NY
C     NX-NUMBER OF STATES (MAX 64 SEE SECOND DIMENSION STATEMENT)
C     NU-NUMBER OF INPUTS (NO MAX)
C     NY-NUMBER OF OUTPUTS (NO MAX). PUT NY=0 IF COMPUTATION OF Y(T)
C        SHOULD BE SKIPPED.
C
C     XO-DUMMY VECTOR OF DIMENSION NX CONTAINING THE
C        THE NEW STATE VECTOR X(T+1). STORED IN DUM8 OF
C        THE COMMON BLOCK /SLASK/. THE FIRST 896 CELLS OF /SLASK/
C        ARE NOT USED.
C
C     SUBROUTINES REQUIRED
C             NONE
C
      DIMENSION A(1,1),B(1,1),C(1,1),X(1),U(1),Y(1)
      DIMENSION XO(64),DUMY(448)
C
      COMMON /SLASK/ DUMY,XO
C
      DO 10 I=1,NX
10    XO(I)=SCAPRO(A(I,1),NX,X(1),1,NX)+SCAPRO(B(I,1),NX,U(1),1,NU)
      CALL MOVE (XO(1),X(1),NX+NX)
C
      IF(NY) 99,99,12
12    DO 14 I=1,NY
14    Y(I)=SCAPRO(C(I,1),NY,X(1),1,NX)
C
99    RETURN
      END
      SUBROUTINE GLIRE1(A,BK,C,AL,X,U,Y,NX,NU,NY)
C
C     COMPUTES THE NEW STATE ESTIMATE XE(T+1) AND THE NEW CONTROL
C     U(T+1) FROM XE(T) AND Y(T) FOR THE GENERAL LINEAR REGULATOR
C             XE(T+1)=A*XE(T)+B*U(T)+K*(Y(T)-C*XE(T))
C             U(T+1)=-L*XE(T+1)
C
C     CAN ALSO BE USED AS A PURE KALMAN FILTER BY PUTTING NU=0
C
C     AUTHOR KJ ASTROM 1971-07-22
C
C     A- MATRIX OF ORDER NX*NX
C     BK-MATRIX OF ORDER NX*(NU+NY) CONCATENATION OF B AND K
C     C- MATRIX OF ORDER NY*NX
C     AL-GAIN MATRIX -L OF ORDER NU*NX. NOTICE -L.
C     X- STATE ESTIMATE XE(T) OF DIMENSION NX, RETURNED AS XE(T+1)
C     U- CONTROL VECTOR OF DIMENSION NU, RETURNED AS THE NEW CONTROL
C     Y- MEASUREMENT VECTOR OF ORDER NY
C     NX-NUMBER OF STATES (MAX 64 GLISY )
C     NU-NUMBER OF CONTROL VARIABLES, NU=0 IF PURE KALMAN FILTER
C     NY-NUMBER OF MEASUREMENTS (MAX(NU+NY)=64)
C
C     UE-DUMMY VECTOR OF DIMENSION NU+NY CONCATENATION OF U
C        AND Y-C*XE STORED IN DUM7 OF THE COMMON BLOCK /SLASK/.
C        THE LAST FIELD DUM8 OF /SLASK/ CONTAINS XE AND THE FIRST
C        768 CELLS ARE NOT USED.
C
C     SUBROUTINES REQUIRED
C             GLISY
C
      DIMENSION A(1,1),BK(1,1),C(1,1),AL(1,1),X(1),U(1),Y(1)
      DIMENSION UE(64),DUMY(384),DUM8(64)
C
      COMMON /SLASK/DUMY,UE,DUM8
C
      IF(NU) 10,10,11
11    CALL MOVE(U(1),UE(1),NU+NU)
10    DO 12 I=1,NY
      I1=NU+I
12    UE(I1)=Y(I)-SCAPRO(C(I,1),NY,X(1),1,NX)
      CALL GLISY1(A,BK,AL,X,UE,U,NX,NU+NY,NU)
C
      RETURN
      END
```