On a Production Planning Problem

Sternby, Jan

1972

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

ON A PRODUCTION PLANNING PROBLEM

JAN STERNBY

Report 7229 december 1972
Lund Institute of Technology
Division of Automatic Control

ABSTRACT.

In this report a production planning problem for a paper factory is discussed. The main difficulty is due to an integer valued criterion, namely the number of production rate changes, that should be minimized. Some possible ways of solving the problem are considered. A method based on non-linear function optimization has been developed and successfully used on a couple of test examples. Finally, a reduced problem is presented, which contains the main difficulties.

TABLE OF CONTENTS

1. INTRODUCTION.

In this report a problem of production planning for a paper factory is discussed. B. Pettersson [1] has previously treated the same problem, and it is assumed that the reader of this report has also access to [1].

A paper mill consists of a network of processing units. An example of this is the Gruvön mill, a model of which is shown in Fig. 2.1 on page 4 in [1].

If the rate of producing paper is determined in advance, then the production rate of the processing units must be set so that the storage tanks are not overfilled or emptied. Some processes consume steam, others produce it, and also some extra steam can be produced. Total production and consumption of steam must be equal.

The problem now is to determine the production rates over a specified planning period so that all restrictions are fulfilled and certain criterias are optimized. The main difficulty is one of the criterias, namely that the <u>number</u> of production rate changes should be as small as possible.

Sometimes it is known in advance that during a certain time some process must be shut down for some reason. Then if this process is steam-producing we would like to store steam before the stop to use during the interrupt, but this is impossible. However, we can do it indirectly for instance by specifying suitably chosen tank levels at the end of the preceding planning period.

In [1] this problem is studied for a paper mill with 3 paper machines, 9 processing units and 10 storage tanks. The model used in [1] has been taken as a starting point in this report, and a method to get a (frequently non-optimal) solution is presented.

The purpose of this report is to discuss alternative solutions.

In Ch. 2 the problem is stated and in Ch. 3 an immediate result on the solutions is proved. Some different ideas for solving the problem are discussed in Ch. 4, and in Ch. 5 a method is described, followed by a couple of test examples in Ch. 6. During the work some new interesting questions have arisen and are discussed in Ch. 7. The problem is finally reduced in Ch. 8, but still it is difficult.

2. STATEMENT OF THE PROBLEM.

Notations:

$x(t)$	vector of storage tank levels
$u(t)$	vector of production rates for processing units
$v(t)$	vector of paper production rates
$S(t)$	extra steam production
T	length of the planning period

With these notations the model for a paper mill as taken from [1] is

$$\frac{dx(t)}{dt} = Bu(t) + Cv(t) \qquad x(0) \text{ given}$$

$$S(t) = Du(t) + Ev(t)$$

$$u_j^{min} \leq u_j(t) \leq u_j^{max} \qquad j = 1, \ldots, n \qquad (*)$$

$$x_i^{min} \leq x_i(t) \leq x_i^{max} \qquad i = 1, \ldots, n$$

$$S^{min} \leq S(t) \leq S^{max}$$

$$v(t) \text{ given for } 0 \leq t \leq T$$

B,C,D and E are matrices of appropriate dimensions, and n is the number of tanks.

<u>Problem</u>: Find a stepwise constant $u(t)$, $0 \leq t \leq T$ fulfilling the above restrictions such that

a) there are as few changes in production rate as possible (where two components of u changing simultaneously are counted as two changes),

b) it is possible to store steam indirectly,

c) final tank levels $x(T)$ are acceptable.

Remarks:

- In this report the functions $v(t)$ are step-wise constant just as in [1]. Of course, this restriction is not necessary for the problem to be relevant, but makes it easier to solve (and is true in the paper machine application).

- There is no dependance of x in the right member of (*). According to [1] this is correct enough for a paper mill. Inclusion of a term Ax in the right member would make the problem more difficult but still sensible.

- The model in [1] is not controllable because there are 10 tanks but only 9 processes. However, for a real mill it should be possible to cut down the number of equations (= number of tanks) so that it is, since this only means that the contents of the extra tanks are determined by the contents of the other tanks.

- Both in [1] and here the objectives b) and c) are taken care of by specifying the final state $x(T)$. This may be difficult to do, as the value of $x(T)$ might influence the minimum number of changes in u (see further Ch. 7).

3. AN IMMEDIATE CONCLUSION.

It is easy to get an upper bound on the number of changes. To see this assume that $v(t)$ is constant in the interval $(0,T)$ and $x(T)$ is reached by applying $u'(t)$ that changes at times $t_1, \ldots, t_{n-1}$. Put $t_0 = 0$, $t_n = T$ and $u'(t) = u_j$ for $t_{j-1} \leq t < t_j$, $j = 1, \ldots, n$. $x(T)$ is then also reached by

$$u''(t) = \frac{1}{T} \sum_{j=1}^{n} (t_j - t_{j-1})u_j$$

and this constant u is an allowed one because

i) $x(t)$ will stay within its bounds since $x(0)$ and $x(T)$ are allowed and $x(t)$ for $0 \leq t \leq T$ is in between as we apply a constant u.

ii) $u''(t)$ is within the boundaries since it is by construction smaller than the biggest u_j and bigger than the smallest one, and these two are allowed.

iii) Let $S'' = Du'' + Ev$ and $S_j = Du_j + Ev$. Then we have

$$S'' = \frac{1}{T} \sum_{j=1}^{n} (t_j - t_{j-1})S_j$$

and analogously to ii) S'' is allowed.

This shows that a constant u is sufficient as long as $v(t)$ is constant. Now suppose that $v(t)$ changes at times $t_1, \ldots, t_k$, and $x(T)$ is reached by some u. Then this u will give certain $x(t_1), x(t_2), \ldots, x(t_k)$ that all are allowed. But we have already shown that a constant u suffices in each of these intervals and hence the minimal number of changes is less than n x (number of changes in $v(t)$) where n is the number of u:s.

Remark: If we want to minimize the number of changes in u it is generally not enough to change u when v changes (see Ch. 8).

4. SOME POSSIBLE METHODS.

We can classify the problem as an optimization problem with a number of restrictions of which some are nonlinear. The major difficulty is that we want to minimize the <u>number</u> of changes in u. The restriction on the steam production, S, is also rather difficult to handle, since it acts as a time-varying restriction on u, that changes when v(t) does. Moreover, this restriction seems to be active in many cases (see e.g. the planning examples on page 65 in [1]).

Sometimes a feasible way of solving optimization problems is to look for candidates, satisfying some necessary conditions for optimum. But such conditions are very hard to find in this case. Also the solutions are mostly not unique (see e.g. Fig. 7.7 on page 50 in [1]), but there is a whole set of u:s giving the same number of production rate changes. It is easily realized, that the set of feasible u:s making steps at the same time is always convex.

4.1. Simulation.

Simulation is a useful tool to get insight into the behaviour of the model, but will not solve the problem for an arbitrary initial state since it is very time-consuming and there is no way of knowing when a solution is optimal.

4.2. Linear Programming.

The problem seems linear, but how choose the objective function? In [1] two different objective functions have been tested:

a) the sum of <u>magnitudes</u> of changes in u,

b) deviation from the desired final state.

As expected, in the first case many of the tanks were left completely empty or filled up and in the second case there were a lot of changes in u. A natural thing to do would be to minimize the sum of magnitudes under the restriction that x(T) should be as desired, but this has not been tried because

i) The method does not minimize the <u>number</u> of changes in u, and there is a big risk that the resulting u will have a lot of small changes.

ii) In [1] only a few fixed times t have been allowed for changes in u. It would be nice to be able to regard these times as variables. This introduces, however, a nonlinearity of the form u x t when calculating the x:s.

iii) The number of variables and restrictions is rather big, which makes the execution time long.

4.3. Minimizing a Performance Functional.

This method is used in [1], where some different possible loss functionals are considered. Unfortunately no one is really minimizing the number of changes in u.

A way out could be to find a connected problem, possible to solve, whose solution would also be the solution to the original problem. The number of changes in u could for instance be approximated by some real number. It seems, however, difficult to find such a connected problem.

4.4. Dynamic Programming.

As a first attempt we put

$N(x,t)$ = the minimal number of changes in u to drive the system from state x at time t to the desired final state at time T.

Then we have

$$N(x,t) = \min_u \Big| \text{number of u-changes in the interval } (t,t+\Delta t) + N\big(x(t+\Delta t),t+\Delta t\big) \Big|$$

Problem: Are changes at time t included in $N(x,t)$ or not?

I) If they are we must know $u(t-)$ to see if there is a change at time t, so $N(x,t)$ should actually be $N(x,u,t)$.

II) If they are not we must remember the u:s belonging to $\big(x(t+\Delta t),t+\Delta t\big)$. Very often there are a lot

> of u:s that are equally good, and all these u:s must be remembered as we go backwards in time. This does not seem to be a possible method.

There is also the storage problem. 9 tanks, with only 3 levels, give $3^9 \approx 20{,}000$ combinations, each with a lot of good 9-dimensional u:s.

Another problem is the choosing of times t and Δt for the Dynamic Programming. At this point Dynamic Programming was abandoned.

4.5. Splines.

The problem can also be regarded as a problem of function approximation. To see this it is instructive to split up the x into two parts, x_1 and x_2. For a one-dimensional case we have

$$\frac{dx}{dt} = bu + cv$$

where we put $x = x_1 - x_2$ with

$$\frac{dx_1}{dt} = bu \qquad \text{and} \qquad \frac{dx_2}{dt} = -\,cv$$

Since v is known in advance $x_2(t)$ can be drawn in a diagram versus time, where for simplicity $x_2(0)$ is set to zero. Now $x_1(0)$ is determined by x(0) and the problem can be written:

> Find a stepwise constant admissible u with as few changes as possible so that x_1 approximates x_2 well enough.

A u is admissible if both S and u are within their respective boundaries at all times. Since x is the difference between x_1 and x_2 the approximation is good enough if x_1 never differs from x_2 by more than a certain amount, equal to the boundary on x. This means that x_1 must stay in the tunnel formed by the dotted lines of Fig. 1.

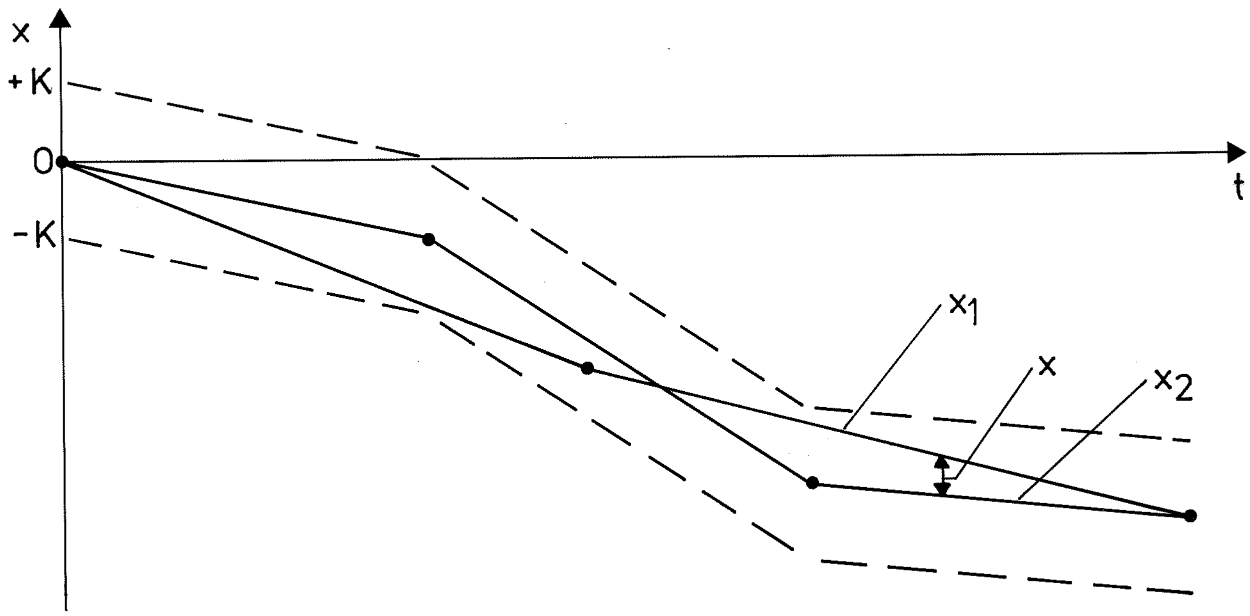

Fig. 1.

Here the boundaries on x have been set to $|x| \leq K$, and no restrictions have been made on u or S. Now the problem could be formulated as approximating a given spline of degree 1 well enough by another spline of degree 1 with as few knots as possible. (Note that the endpoints of x_1 are fixed.)

Unfortunately, the theory for this is not quite developed. Moreover, the situation is complicated by the restrictions on u and S, i.e. restrictions on the slopes of the splines. Also, when x is ten-dimensional there will be ten curves to approximate, and the different x_1-curves will be linked together by the matrix B.

This way of looking at the problem does not solve it, but gives the idea to the method proposed in Ch. 5.

It might be good to split up the problem and first calculate the minimal number of changes, then which u:s should change and finally the times for change and the resulting u-values. But no way has been found to do this except trying, which is utilized in the method described in the next chapter.

5. A FUNCTION MINIMIZATION METHOD.

The minimization of the number of changes in u is done by sequentially searching solutions with 0,1,2,... changes. Each search is done by minimizing a function $f(u_1, u_2, \ldots, u_\ell, t_1, t_2, \ldots, t_\ell)$ that punishes every limit exceeded. When this function becomes zero we have found an admissible solution.

The arguments in f are $t_1, \ldots, t_\ell$, the times for changes in u and $u_1, \ldots, u_\ell$, the new values of the components changed. In f is also included a quadratic punishment on the t:s falling outside the interval (0,T).

Since there is no dependance on x in the right member of (*) we know that if x, u and S are within the limits at the times where u or v makes a step, then they will always be. For these times we just calculate the x and if some component of x or u are out of limits a punishment is added to $f(u_1, \ldots, u_\ell, t_1, \ldots, t_\ell)$, proportional to the square of the exceeding amount. For S the punishment is also multiplied by the square of the duration of the excession (in order to make f smoother near the minimum point).

The great advantage with this loss function is that it is zero for an admissible solution, which means that these are easily recognized. There are, however, some problems. First of all, for every minimization of f we have to decide in advance which components of u that are going to change. Since this is generally not known the best thing would be to try all combinations of changes. But distributing for instance 4 changes to 9 u:s gives 495 possibilities, so it is necessary to reduce the number of minimizations. Here this is done in the following way. If there is no admissible solution with zero or one change, then one change is fixed to the u-component giving the smallest loss function. (Time and magnitude of the change is still free.) A second change is now tried on one u-component after the other, and if still no admissible solution is found we fix the second change on the same grounds as the first one and go on. This method will probably not give the minimum number of changes in general, but has given a solution to some test examples. It is possible that another loss function, e.g. the maximal time before some limit is exceeded, is better, but this has not been tested. It would, of course, be nice, but seems difficult, to prove either that the method does give the minimum number of changes or the contrary.

Another problem is the possible existence of more than one local minimum point for f. In such a case the numerical minimization could arrive at a local minimum point instead of the global one, and this might destroy the method. If f is such, again another f could possibly be better.

For further details about the loss function see the Appendix, where the FORTRAN programs needed are also given.

6. TEST EXAMPLES.

Example 1: The first example is a two-dimensional one. Using the notations of (*) we have with T = 5

$$B = \begin{vmatrix} 1 & 0 \\ 1 & -1 \end{vmatrix} \qquad C = \begin{vmatrix} -1 \\ 0 \end{vmatrix} \qquad D = [1 \quad -2] \qquad E = [2]$$

$$x(0) = \begin{vmatrix} 0.5 \\ 0.5 \end{vmatrix} \qquad x(5) = \begin{vmatrix} 0.5 \\ 0.5 \end{vmatrix}$$

$$0.5 \leq u_1 \leq 1 \qquad 0.2 \leq u_2 \leq 1$$

$$0 \leq S \leq 2$$

$$0 \leq x_1 \leq 1 \qquad 0 \leq x_2 \leq 1$$

$$V(t) = \begin{cases} 0.9 & 0 \leq t < 2 \\ 0 & 2 \leq t < 3 \\ 0.9 & 3 \leq t \leq 5 \end{cases}$$

This problem has also been solved in [1] (page 37), but since $x_3 = 1 - x_2$ there, x_3 has not been included here. The first change was fixed to u_2 since the loss function was 0.0595 for u_1 and 0.0172 for u_2. A second change on u_2 then solved the problem just as in [1], where it is also shown that 2 is the minimum number of changes.

The solutions obtained with the final method of [1] (Fig. 7.4) and with the method of Ch. 5 in this report are shown in Fig. 2 and Fig. 3. The difference is that in the latter case u_2 does not change the second time until $t = 3.13$, which makes x_2 go further to the limit. However, this change can be moved to $t = 3$.

Example 2: The second problem is nearly identical to "planning example 1" in [1]. We have

$T = 48, \quad x_i(0) = x_i(48) = 50\% \quad \text{for} \quad i = 1, \ldots, 9$

$15\% \leq x_i(t) \leq 85\% \qquad i = 1, \ldots, 9$

In [1] there are 10 tanks x, but only 9 processes u, so it is not possible to control all final tank levels.

$x_{10}(48)$ is not reached in [1], and therefore x_{10} is not at all taken into account here.

To this problem was found a solution with only one change, placed on u_7 at time 23.36 (can be moved to 24.00), while in [1] there are two changes on u_7 at times 24 and 32.

These two solutions are shown in Fig. 4 and Fig. 5. To get only one change we have to utilize especially the capacity of x_7 harder. x_6, S and, of course, u_7 are also changed.

The two other planning examples in [1] have not been solved since they are not solvable with the given specifications on x(48).

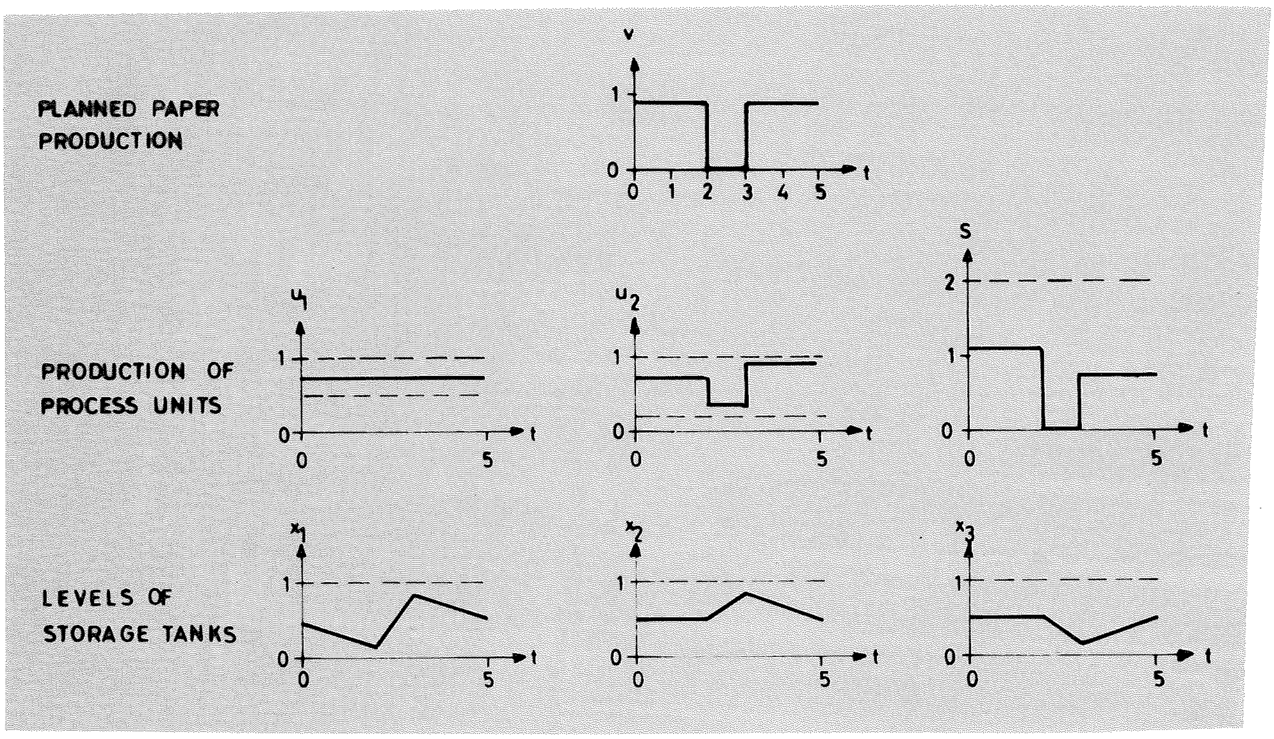

Fig. 2 - Solution to Example 1 as given on page 47 in [1].

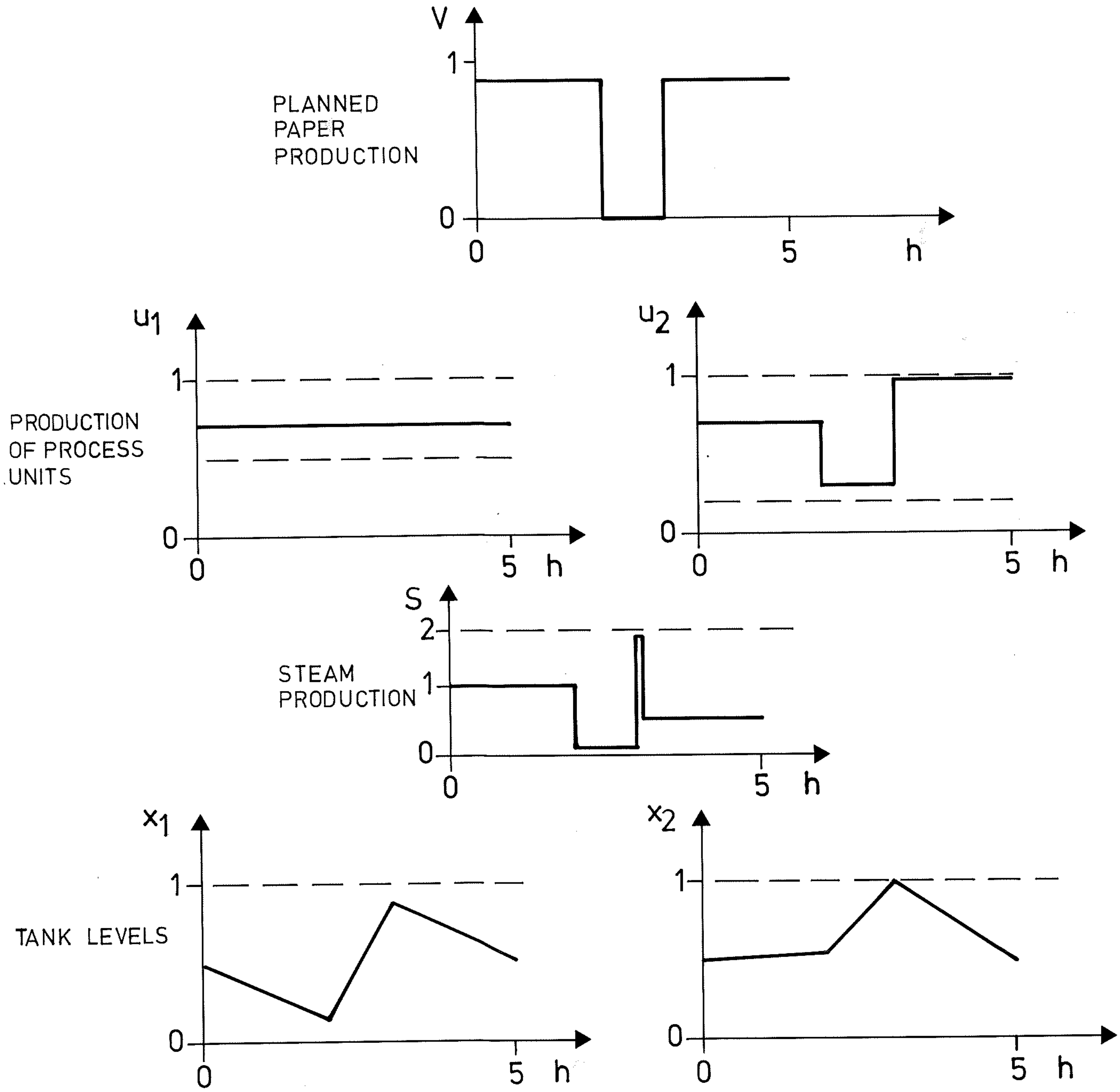

Fig. 3 - Solution to Example 1 using the method of Ch. 5 in this report.

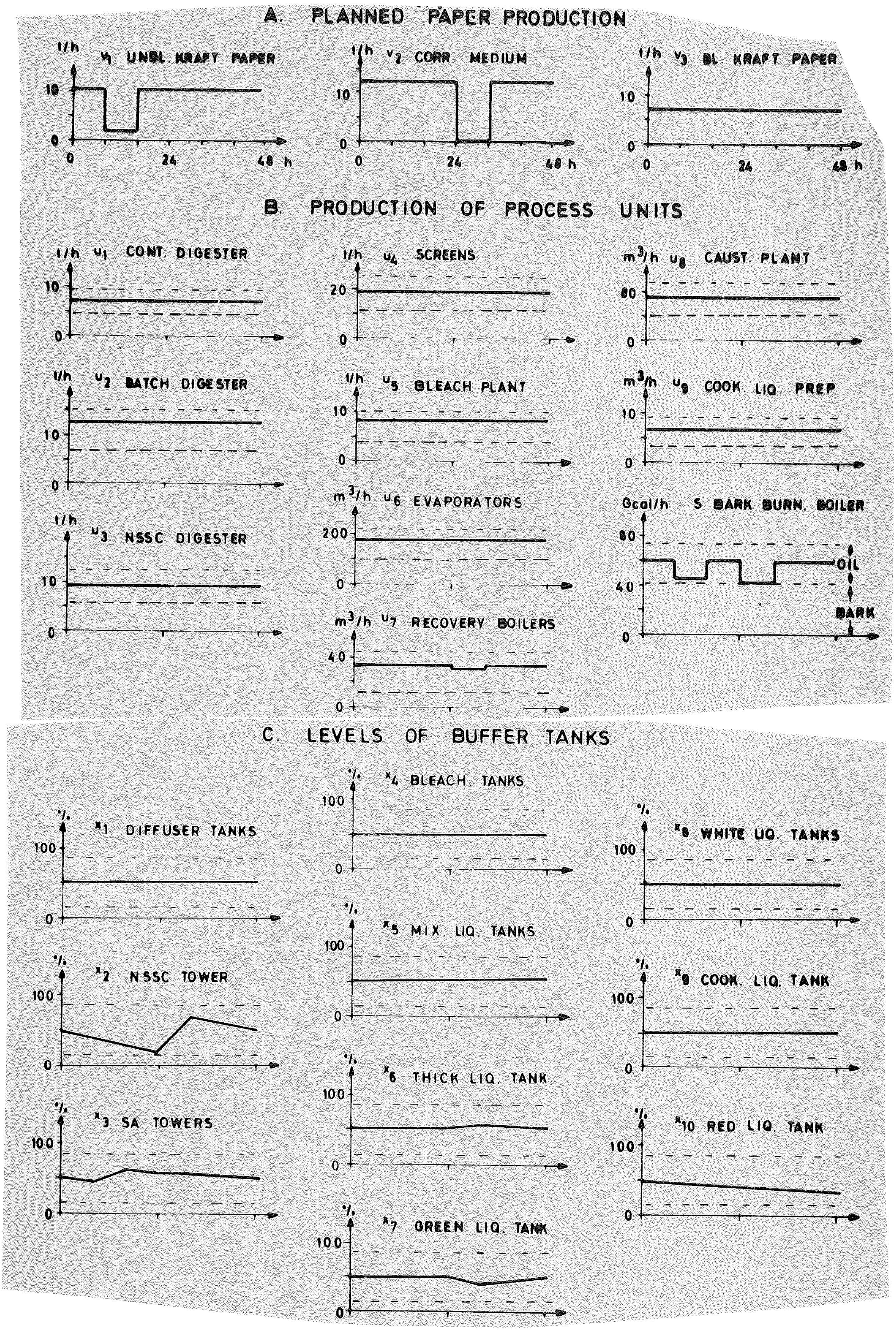

Fig. 4 - The solution obtained in [1] to Example 2. This figure is identical to Fig. 9.1 on page 67 in [1].

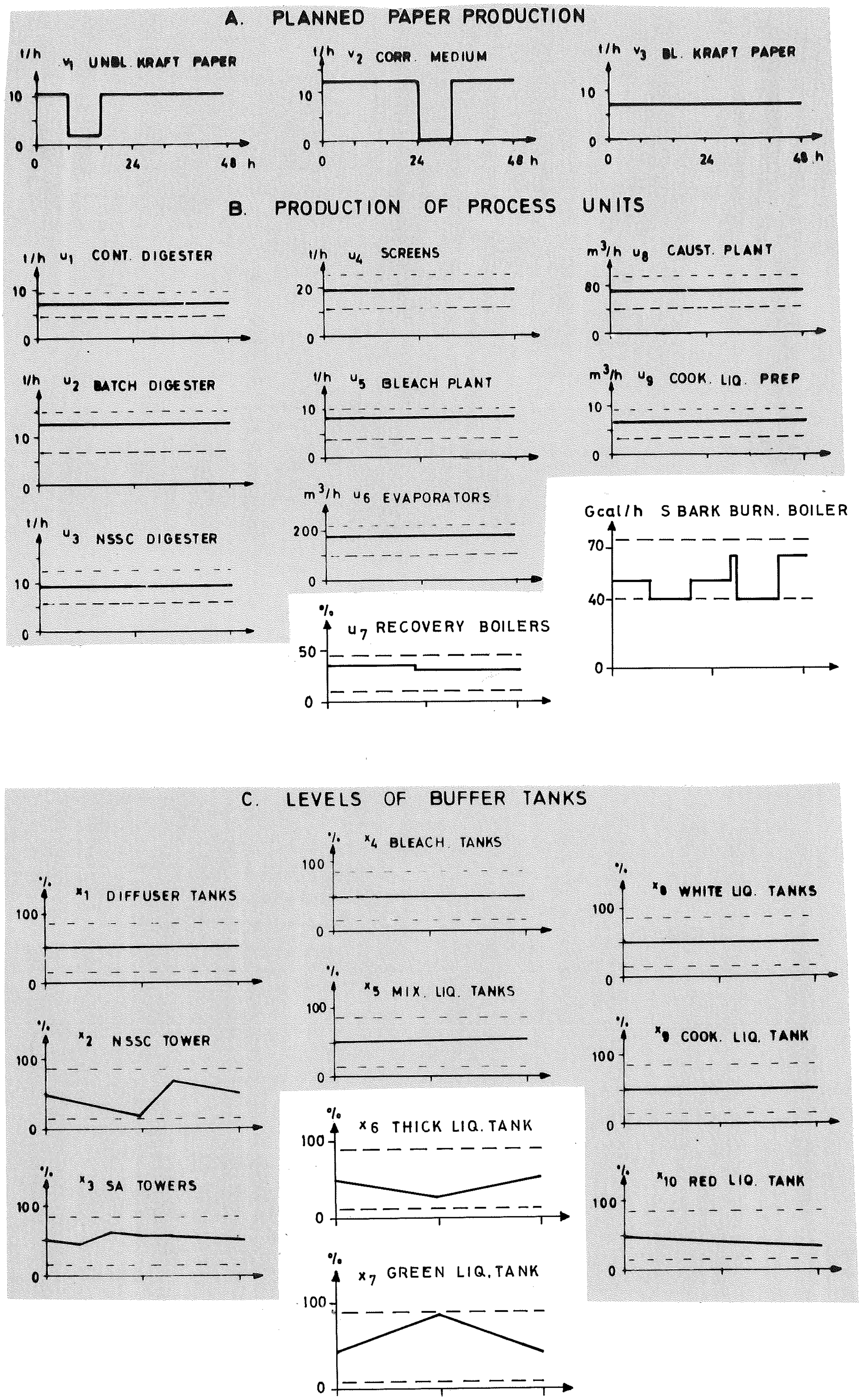

Fig. 5 - A solution to Example 2 obtained by the method of Ch. 5.

7. SOME FURTHER PRACTICAL ASPECTS.

If it is necessary to get some solution to a problem which is unsolvable because of the specifications on $x(T)$, then we must have a systematic method for changing $x(T)$. Otherwise there is no guarantee at all that the new $x(T)$ will lead to a solvable problem. In fact, an interesting question is: What $x(T)$ can be reached? A partial answer is that the reachable $x(T)$:s form a convex set. This can be seen in the following way. Assume that x' and x'' are both reachable and the corresponding u:s are u' and u''. Then $\lambda x' + (1-\lambda)x''$ is reached by applying $\lambda u' + (1-\lambda)u''$. That this u and all intermediate x:s are allowed is easily checked.

Also it seems very probable that in many cases the minimum number of changes in u is very strongly dependant on the $x(T)$ specified. If then the exact value of $x(T)$ is not so important it would be better not to specify it as a <u>fixed</u> vector of numbers.

In practice T is about 48 hours for the paper mill at Billerud. Then the planning has to be re-done every second day. This means that every second day, when a new planning period starts, all the u:s are changed. An interesting problem would be to try minimizing some <u>total</u> number of u-changes.

Sometimes the paper machine stops by e.g. a paper break. It is desirable to include also these unplanned stops in the model, for instance by some statistical method.

8. A SIMPLE PROBLEM.

The essential difficulty in counting the number of u-changes is still left if the problem is reduced into a one-dimensional one. Consider the following

<u>Problem formulation</u>.

$$\frac{dx}{dt} = u + v \qquad -1 \leq x \leq 1 \qquad (**)$$

$x(0)$, $x(1)$ and $v(t)$ for $0 \leq t \leq 1$ given, where v is stepwise constant.

Determine a $u(t)$ for $0 \leq t \leq 1$, stepwise constant with as few steps as possible, that satisfies (**).

As there are no limitations on u all $x(1)$ that are allowed will also be possible to reach.

Two difficulties from the multi-dimensional case have disappeared here. First of all, there is no question now of which u-component that should change. Secondly the restriction on S has been taken away. (Since S contains the v:s it acts as a time-varying restriction on u.)

There are at least two ways of looking at this problem geometrically, the first one being the separation of x into two parts described in Section 4.5 about splines.

<u>Example</u>: Let

$$T = 7, \quad x(0) = x(7) = 0, \quad -1 \leq x(t) \leq 1$$

$$v(t) = \begin{cases} 0 & 0 \leq t < 1 \\ 3 & 1 \leq t < 2 \\ 0.5 & 2 \leq t < 4 \\ 3 & 4 \leq t < 6 \\ 1 & 6 \leq t < 7 \end{cases}$$

The resulting curves for x_1 and x_2 are shown in Fig. 6. The x_1-curve shown is the only possible one with one change in u, so in this example u should not change when v does and not when x hits a limit.

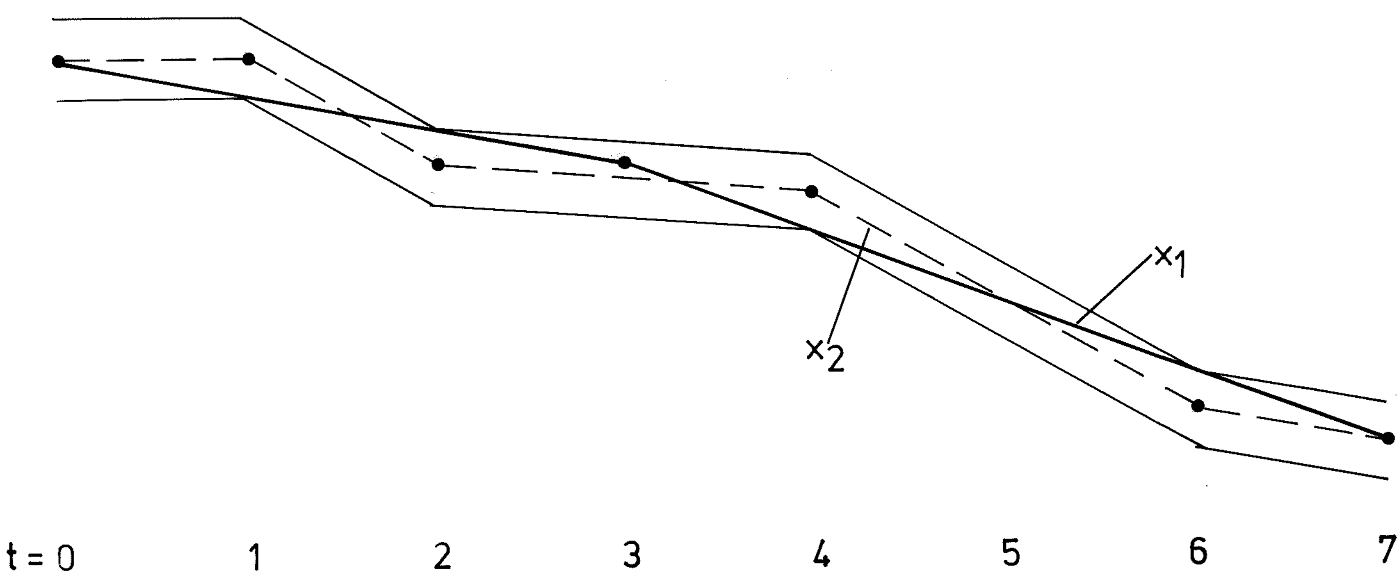

Fig. 6.

The second way to look at the problem geometrically is to calculate the constant u needed to bring x from x(0) to x(T), then apply this u and look at the curve for x(t). A change at time t_1 then means a change of the slope, so that x(0) and x(T) are not changed, $x(t_1)$ gets the biggest change and x(t) will change proportionally to t:s distance from 0 (T) if $t < t_1$ ($t > t_1$). The x-curve for no changes is shown in Fig. 7 for the previous example.

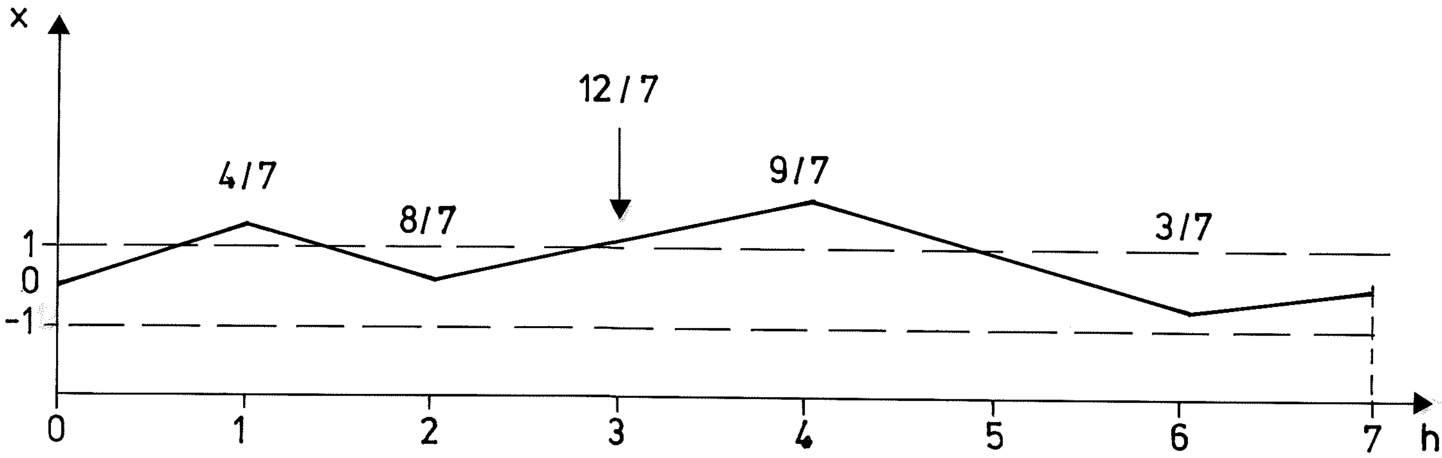

Fig. 7.

The dotted lines are the limits for x. One change should be applied at the arrow and should decrease that point by 12/7. Then the corners will decrease by the amounts written above, and they will all touch the limits.

In Ch. 3 was shown that the minimal number of changes is less than the number of changes in v (when we have only one u). We can also get a lower bound. Start from $t = 0$. When $x(t)$ crosses the boundary for the first time it means that we must have at least one change in u. Each time $x(t)$ then goes through all of the allowed area and crosses the other boundary the least number of changes will be increased by one. For the example in this chapter we have

$$1 \leq \text{min. number of changes} \leq 4$$

It is probably necessary to find a good method for solving the reduced problem described in this chapter before the real problem can be solved.

8. REFERENCE.

[1] Pettersson, B.: Mathematical Methods of a Pulp and Paper Mill Scheduling Problem, Report 7001, April, 1970, Div. of Automatic Control, Lund Institute of Technology.

APPENDIX

To carry out the minimizations of Ch. 5 we use a set of FORTRAN programs described below. In the head program, JANS, the subroutine INLAS is called first. INLAS reads all data needed about the system into two common areas, and prints it again. Then a subroutine DECOM is called (see below) and some necessary parameters for the subroutine POWEL are set. POWEL performs the minimization and then the result is reorganized and put into the subroutine BIINT, that integrates the system and presents all interesting data about the solution.

POWEL uses the subroutine FUNC to calculate function values during the minimization. First FUNC takes the vector NOD from the common area. This vector describes what components of u should change. By calling the subroutine SRT the changes are sorted in time-order (to simplify the other programs). Then the subroutine SUB1 is called, and calculates the initial u:s needed to drive the system from the given x(0) to x(T).

Now the system is integrated by the function F. F is returned containing quadratic punishments on all x:s exceeding the limits. The function BIVIL calculates the punishments on S and u, and the loss function is set to F+CSTR*BIVIL. In the test examples of this report CSTR has been set equal to one. Finally a very heavy punishment is added on the t:s falling outside the interval $(10^{-7}*T,T)$.

In order to calculate the initial u:s in SUB1 an equation system has to be solved. To do this we use the subroutines DECOM and SOLVB. In DECOM a decomposition of the coefficient matrix is performed. This has to be done only once, so DECOM is called from the head program in order to save computing time.

All the programs are listed below starting with the head program except DECOM, SOLVB and POWEL which belong to the program library of the Division of Automatic Control.

```
C     THIS PROGRAM READS EVERYTHING IN COMMON (SEE SUB1) AND COMMON/BIVI/ (SEE
C     BIVIL) AND STARTING VALUES OF X  (U AND CORRESPONDING T) . IT MINIMIZES
C     FUNC(MR,X,F) WITH POWEL AND PRINTS OUT MINIMUM F-VALUE, THE RESULTING U:S
C     AND WHEN THEY CHANGE.
C
C     REMARKS     NOM AND NV MUST NOT BE ZERO
C                 NTV MUXT NOT BE ZERO OR ONE
C                 NNU MUST BE SET TO THE DECLARED DIMENSION OF B
C
C     SUBROUTINES REQUIRED               DECOM
C                                        POWEL
C                                        SRT
C                                        SUB1
C                                        FUNC
C                                        INLAS
C                                        BIINT
C                                        (SOLVB)
C                                         (F)
C                                        (BIVIL)
C
C
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      COMMON /BIVI/ D(10),E(10),UMIN(10),UMAX(10),SMAX,SMIN
      DIMENSION G(20),X(20),U(20),NO(10),T(10)
      EXTERNAL FUNC
      NNU=10
      EPS=1.E-07
      CALL INLAS
      MR=2*NOM
      READ 200,(X(I),I=1,MR)
  200 FORMAT(5E16.8)
C
      CALL DECOM(B,NUU,NNU,EPS,ISING)
      IF (ISING) 4,4,6
    6 PRINT 250,ISING
  250 FORMAT(10X,'DECOM GIVES ISING=',I2)
      GO TO 40
C
    4 DO  5 I=1,MR
    5 G(I) = 1.E-04
      ESCALE = 500000.
      T(NOM+1)=TSLUT
      IPRINT = 3
      MAXIT = 100
      ICON = 1
      CALL POWEL(X,G,MR,F,ESCALE,IPRINT,ICON,MAXIT,FUNC)
C
      PRINT 300,F
  300 FORMAT(/10X,'FUNCTION VALUE',E16.8)
      DO 10 I=1,NOM
      U(I)=X(I)
      T(I)=X(I+NOM)
   10 NO(I)=NOD(I)
      CALL SRT(U,NO,T,NOM)
      DO 20 I=1,NOM
      K=NOM-I+1
   20 U(K+NUU)=U(K)
      CALL SUB1(U,NO,T)
      CALL BIINT(U,NO,T)
   40 CONTINUE
      STOP
      END
```

```
C     THIS PROGRAM READS EVERYTHING IN COMMON (SEE SUB1) AND COMMON/BIVI/ (SEE
C     BIVIL) AND STARTING VALUES OF X  (U AND CORRESPONDING T) . IT MINIMIZES
C     FUNC(MR,X,F) WITH POWEL AND PRINTS OUT MINIMUM F-VALUE, THE RESULTING U:S
C     AND WHEN THEY CHANGE.
C
C     REMARKS       NOM AND NV MUST NOT BE ZERO
C                   NTV MUXT NOT BE ZERO OR ONE
C                   NNU MUST BE SET TO THE DECLARED DIMENSION OF B
C
C     SUBROUTINES REQUIRED                       DECOM
C                                                POWEL
C                                                SRT
C                                                SUB1
C                                                FUNC
C                                                INLAS
C                                                BIINT
C                                                (SOLVB)
C                                                 (F)
C                                                (BIVIL)
C
C
C
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      COMMON /BIVI/ D(10),E(10),UMIN(10),UMAX(10),SMAX,SMIN
      DIMENSION G(20),X(20),U(20),NO(10),T(10)
      EXTERNAL FUNC
      NNU=10
      EPS=1.E-07
      CALL INLAS
      MR=2*NOM
      READ 200,(X(I),I=1,MR)
  200 FORMAT(5E16.8)
C
      CALL DECOM(B,NUU,NNU,EPS,ISING)
      IF (ISING) 4,4,6
    6 PRINT 250,ISING
  250 FORMAT(10X,'DECOM GIVES ISING=',I2)
      GO TO 40
C
    4 DO  5 I=1,MR
    5 G(I) = 1.E-04
      ESCALE = 500000.
      T(NOM+1)=TSLUT
      IPRINT = 3
      MAXIT = 100
      ICON = 1
      CALL POWEL(X,G,MR,F,ESCALE,IPRINT,ICON,MAXIT,FUNC)
C
      PRINT 300,F
  300 FORMAT(/10X,'FUNCTION VALUE',E16.8)
      DO 10 I=1,NOM
      U(I)=X(I)
      T(I)=X(I+NOM)
   10 NO(I)=NOD(I)
      CALL SRT(U,NO,T,NOM)
      DO 20 I=1,NOM
      K=NOM-I+1
   20 U(K+NUU)=U(K)
      CALL SUB1(U,NO,T)
      CALL BIINT(U,NO,T)
   40 CONTINUE
      STOP
      END
```

```
      SUBROUTINE INLAS
C
C
C     THIS ROUTINE READS  ALL VARIABLES IN BLANK COMMON AND IN COMMON
C      /BIVI/ AND PRINTS THEM  (SEE SUB1 AND BIVIL )
C
C     NO SUBOUTINES REQUIRED
C
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      COMMON /BIVI/ D(10),E(10),UMIN(10),UMAX(10),SMAX,SMIN
      READ 100,N
      READ 100,NUU
      READ 100,NV
      READ 100,NTV
      READ 100,NOM
      DO 1  I=1,N
    1 READ 200,(B(I,J),J=1,NUU)
      DO 2  I=1,N
    2 READ 200,(C(I,J),J=1,NV)
      READ 200,(D(I),I=1,NUU)
      READ 200,(E(I),I=1,NV)
      READ 200,(X0(I),I=1,N)
      READ 200,(XT(I),I=1,N)
      READ 200,(TV(I),I=1,NTV)
      DO  3 I=1,NV
    3 READ 200,(V(I,J),J=1,NTV)
      READ 100,(NOD(I),I=1,NOM)
      READ 200,TSLUT
      READ 200,XMAX
      READ 200,SMAX
      READ 200,SMIN
      READ 200,CSTR
      READ 200,(UMIN(I),I=1,NUU)
      READ 200,(UMAX(I),I=1,NUU)
      READ 200,(XBEG(I),I=1,N)
      READ 200,(XTBEG(I),I=1,N)
  100 FORMAT(10I5)
  200 FORMAT(5E16.8)
      DO 20 I=1,N
      SL=XBEG(I)
      DO 10 J=1,NUU
   10 B(I,J)=B(I,J)/SL*100.
      DO 20 J=1,NV
   20 C(I,J)=C(I,J)/SL*100.
      DO 25  I=1,NUU
      SL=UMAX(I)
      DO 30 J=1,N
   30 B(J,I)=B(J,I)*SL/100.
      D(I)=D(I)*SL/SMAX
   25 UMIN(I)=UMIN(I)/SL*100.
      DO 35 I=1,NV
   35 E(I)=E(I)/SMAX*100.
```

```
      SUBROUTINE INLAS
C
C     THIS ROUTINE READS  ALL VARIABLES IN BLANK COMMON AND IN COMMON
C      /BIVI/ AND PRINTS THEM  (SEE SUB1 AND BIVIL )
C
C     NO SUBOUTINES REQUIRED
C
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,XO(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      COMMON /BIVI/ D(10),E(10),UMIN(10),UMAX(10),SMAX,SMIN
      READ 100,N
      READ 100,NUU
      READ 100,NV
      READ 100,NTV
      READ 100,NOM
      DO 1  I=1,N
    1 READ 200,(B(I,J),J=1,NUU)
      DO 2  I=1,N
    2 READ 200,(C(I,J),J=1,NV)
      READ 200,(D(I),I=1,NUU)
      READ 200,(E(I),I=1,NV)
      READ 200,(X0(I),I=1,N)
      READ 200,(XT(I),I=1,N)
      READ 200,(TV(I),I=1,NTV)
      DO  3 I=1,NV
    3 READ 200,(V(I,J),J=1,NTV)
      READ 100,(NOD(I),I=1,NOM)
      READ 200,TSLUT
      READ 200,XMAX
      READ 200,SMAX
      READ 200,SMIN
      READ 200,CSTR
      READ 200,(UMIN(I),I=1,NUU)
      READ 200,(UMAX(I),I=1,NUU)
      READ 200,(XBEG(I),I=1,N)
      READ 200,(XTBEG(I),I=1,N)
  100 FORMAT(10I5)
  200 FORMAT(5E16.8)
      DO 20 I=1,N
      SL=XBEG(I)
      DO 10 J=1,NUU
   10 B(I,J)=B(I,J)/SL*100.
      DO 20 J=1,NV
   20 C(I,J)=C(I,J)/SL*100.
      DO 25  I=1,NUU
      SL=UMAX(I)
      DO 30 J=1,N
   30 B(J,I)=B(J,I)*SL/100.
      D(I)=D(I)*SL/SMAX
   25 UMIN(I)=UMIN(I)/SL*100.
      DO 35 I=1,NV
   35 E(I)=E(I)/SMAX*100.
```

```
      SMIN=SMIN/SMAX*100.
      PRINT 500,N,NUU,NV,NTV,NOM
  500 FORMAT(1H1,36X,'SYSTEM PARAMETERS'///1X,'NUMBER OF  X:S',8X,'U:S',
     B8X,'V:S',2X,'CHANGES IN V CHANGES IN U'/3X,5I11)
      PRINT 501
  501 FORMAT(//36X,'B-MATRIX'/)
      DO 51 I=1,N
   51 PRINT 200,(B(I,J),J=1,NUU)
      PRINT 502
  502 FORMAT(//36X,'C-MATRIX'/)
      DO 52 I=1,N
   52 PRINT 200,(C(I,J),J=1,NV)
      PRINT 503
  503 FORMAT(//36X,'D-MATRIX'/)
      PRINT 200,(D(I),I=1,NUU)
      PRINT 504
  504 FORMAT(//36X,'E-MATRIX'/)
      PRINT 200,(E(I),I=1,NV)
      PRINT 506
  506 FORMAT(//36X,'START VALUE'/)
      PRINT 200,(X0(I),I=1,N)
      PRINT 507
  507 FORMAT(//36X,'FINAL VALUE'/)
      PRINT 200,(XT(I),I=1,N)
      PRINT 508
  508 FORMAT(//36X,'CHANGING TIMES FOR V'/)
      PRINT 200,(TV(I),I=1,NTV)
      PRINT 505
  505 FORMAT(//36X,'V-MATRIX'/)
      DO 53 I=1,NTV
   53 PRINT 200,(V(J,I),J=1,NV)
      PRINT 509
  509 FORMAT(//36X,'NOD-VECTOR '/)
      PRINT 100,(NOD(I),I=1,NOM)
      PRINT 510,TSLUT,XMAX,SMAX,SMIN,CSTR
  510 FORMAT(///36X,' LIMITATIONS '//9X,'TSLUT',10X,'XMAX(PROC.)',6X,'SM
     VAX',12X,'SMIN',12X,'CSTR'/1X,5E16.8)
      PRINT 520
  520 FORMAT(//36X,'UMIN'/)
      PRINT 200,(UMIN(I),I=1,NUU)
      PRINT 521
  521 FORMAT(//36X,'UMAX'/)
      PRINT 200,(UMAX(I),I=1,NUU)
      PRINT 522
  522 FORMAT(//36X,'XBEG(=ACTUAL TANK VOLUME)'/)
      PRINT 200,(XBEG(I),I=1,N)
      PRINT 523
  523 FORMAT(//36X,'XTBEG(PROC.)'/)
      PRINT 200,(XTBEG(I),I=1,N)
      RETURN
      END
```

```
      SMIN=SMIN/SMAX*100.
      PRINT 500,N,NUU,NV,NTV,NOM
  500 FORMAT(1H1,36X,'SYSTEM PARAMETERS'///1X,'NUMBER OF  X:S',8X,'U:S',
     B8X,'V:S',2X,'CHANGES IN V CHANGES IN U'/3X,5I11)
      PRINT 501
  501 FORMAT(//36X,'B-MATRIX'/)
      DO 51 I=1,N
   51 PRINT 200,(B(I,J),J=1,NUU)
      PRINT 502
  502 FORMAT(//36X,'C-MATRIX'/)
      DO 52 I=1,N
   52 PRINT 200,(C(I,J),J=1,NV)
      PRINT 503
  503 FORMAT(//36X,'D-MATRIX'/)
      PRINT 200,(D(I),I=1,NUU)
      PRINT 504
  504 FORMAT(//36X,'E-MATRIX'/)
      PRINT 200,(E(I),I=1,NV)
      PRINT 506
  506 FORMAT(//36X,'START VALUE'/)
      PRINT 200,(X0(I),I=1,N)
      PRINT 507
  507 FORMAT(//36X,'FINAL VALUE'/)
      PRINT 200,(XT(I),I=1,N)
      PRINT 508
  508 FORMAT(//36X,'CHANGING TIMES FOR V'/)
      PRINT 200,(TV(I),I=1,NTV)
      PRINT 505
  505 FORMAT(//36X,'V-MATRIX'/)
      DO 53 I=1,NTV
   53 PRINT 200,(V(J,I),J=1,NV)
      PRINT 509
  509 FORMAT(//36X,'NOD-VECTOR '/)
      PRINT 100,(NOD(I),I=1,NOM)
      PRINT 510,TSLUT,XMAX,SMAX,SMIN,CSTR
  510 FORMAT(///36X,' LIMITATIONS '//9X,'TSLUT',10X,'XMAX(PR6C.)',6X,'SM
     VAX',12X,'SMIN',12X,'CSTR'/1X,5E16.8)
      PRINT 520
  520 FORMAT(//36X,'UMIN'/)
      PRINT 200,(UMIN(I),I=1,NUU)
      PRINT 521
  521 FORMAT(//36X,'UMAX'/)
      PRINT 200,(UMAX(I),I=1,NUU)
      PRINT 522
  522 FORMAT(//36X,'XBEG(=ACTUAL TANK VOLUME)'/)
      PRINT 200,(XBEG(I),I=1,N)
      PRINT 523
  523 FORMAT(//36X,'XTBEG(PROC.)'/)
      PRINT 200,(XTBEG(I),I=1,N)
      RETURN
      END
```

```
      SUBROUTINE BIINT(U,NO,T)
C
C     THIS SUBROUTINE CALCULATES X AND S AT ALL INTERESTING TIMES AND PRINTS OUT
C     ALL U:S , X:S  AND S:S. THE SYSTEM PARAMETERS ARE SUPPLIED BY THE TWO COM-
C     MON AREAS  ( BLANK  AND /BIVI/)
C     NOTATIONS:  SEE SUB1
C
C     NO SUBROUTINE REQUIRED
C
C     REMARK   T(NOM+1) = TSLUT IN THE CALLING PROGRAM
C
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      COMMON /BIVI/ D(10),E(10),UMIN(10),UMAX(10),SMAX,SMIN
      DIMENSION X(10),UN(10),VN(10),U(1),NO(1),T(1)
      PRINT 500
  500 FORMAT(1H1,10X,'PRINTOUT FROM SUBROUTINE BIINT')
C
      DO 5 I=1,10
      X(I)=0.
    5 UN(I)=0.
      IC=0
      DO 10 I=1,N
   10  X(I)=X0(I)
      DO 12 I=1,NUU
   12 UN(I)=U(I)
      DO 15 I=1,NV
   15 VN(I)=V(I,1)
C
      T0=0.
      I=1
      K=1
      TVI=TV(1)
C
C     CALCULATE THE NEXT T.IND TELLS IF  U OR V CHANGES THERE
C
   20 IND=2
      IF(TVI-T(K))35,45,55
   35 TT=TVI
      IND=0
      GO TO 70
   45 IND=1
   55 TT=T(K)
      TT=AMAX1(TT,1.E-07*TSLUT)
C
C     CALCULATE THE NEW X AND CHECK
C
   70 DO 100 J=1,N
      SL1=0.
      SL2=0.
      DO 80 NI=1,NUU
   80 SL1=SL1+B(J,NI)*UN(NI)
      DO 90 NI=1,NV
   90 SL2=SL2+C(J,NI)*VN(NI)
  100 X(J)=X(J)+(SL1+SL2)*(TT-T0)
      T0=TT
      IF(IND-1)102,104,102
  102 IFAKT=1
      GO TO 106
  104 IFAKT=2
  106 IC=IC+IFAKT
      S=0.
      DO 108 J=1,NUU
  108 S=S+D(J)*UN(J)
      DO 110 J=1,NV
  110 S=S+E(J)*VN(J)
      PRINT 200,T0,IND,(X(I),I=1,10),(UN(I),I=1,10),S
  200 FORMAT(//30X,'TIME',G16.6,'  IND=',I2/10X,'X=',5G18.8/12X,5G18.8/1
     V0X,'U=',5G18.8/12X,5G18.8/10X,'S=',G18.8)
```

```
      SUBROUTINE BIINT(U,NO,T)
C
C     THIS SUBROUTINE CALCULATES X AND S AT ALL INTERESTING TIMES AND PRINTS OUT
C     ALL U:S , X:S  AND S:S. THE SYSTEM PARAMETERS ARE SUPPLIED BY THE TWO COM-
C     MON AREAS  ( BLANK  AND /BIVI/)
C     NOTATIONS:  SEE SUB1
C
C     NO SUBROUTINE REQUIRED
C
C     REMARK   T(NOM+1) = TSLUT IN THE CALLING PROGRAM
C
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      COMMON /BIVI/ D(10),E(10),UMIN(10),UMAX(10),SMAX,SMIN
      DIMENSION X(10),UN(10),VN(10),U(1),NO(1),T(1)
      PRINT 500
  500 FORMAT(1H1,10X,'PRINTOUT FROM SUBROUTINE BIINT')
C
      DO 5 I=1,10
      X(I)=0.
    5 UN(I)=0.
      IC=0
      DO 10 I=1,N
   10 X(I)=X0(I)
      DO 12 I=1,NUU
   12 UN(I)=U(I)
      DO 15 I=1,NV
   15 VN(I)=V(I,1)
C
      T0=0.
      I=1
      K=1
      TVI=TV(1)
C
C     CALCULATE THE NEXT T.IND TELLS IF  U OR V CHANGES THERE
C
   20 IND=2
      IF(TVI-T(K))35,45,55
   35 TT=TVI
      IND=0
      GO TO 70
   45 IND=1
   55 TT=T(K)
      TT=AMAX1(TT,1.E-07*TSLUT)
C
C     CALCULATE THE NEW X AND CHECK
C
   70 DO 100 J=1,N
      SL1=0.
      SL2=0.
      DO 80 NI=1,NUU
   80 SL1=SL1+B(J,NI)*UN(NI)
      DO 90 NI=1,NV
   90 SL2=SL2+C(J,NI)*VN(NI)
  100 X(J)=X(J)+(SL1+SL2)*(TT-T0)
      T0=TT
      IF(IND-1)102,104,102
  102 IFAKT=1
      GO TO 106
  104 IFAKT=2
  106 IC=IC+IFAKT
      S=0.
      DO 108 J=1,NUU
  108 S=S+D(J)*UN(J)
      DO 110 J=1,NV
  110 S=S+E(J)*VN(J)
      PRINT 200,T0,IND,(X(I),I=1,10),(UN(I),I=1,10),S
  200 FORMAT(//30X,'TIME',G16.6,'  IND=',I2/10X,'X=',5G18.8/12X,5G18.8/1
     Y0X,'U=',5G18.8/12X,5G18.8/10X,'S=',G18.8)
```

```
C
C     TAKE IN A NEW V OR U OR BOTH DEPENDING ON IND
C
      IF(IC-NTV-NOM)120,150,150
  120 IF(IND-1)130,130,135
  130 I=I+1
      IF(I-1-NTV)132,131,131
  131 TVI=1.E+07*TSLUT
      GO TO 134
  132 DO 133 J=1,NV
  133 VN(J)=V(J,I)
      TVI=TV(I)
  134 IF(IND.EQ.0) GO TO 20
  135 NOCKE=NO(K)
      UN(NOCKE)=U(NUU+K)
      K=K+1
C
      GO TO 20
C
  150 RETURN
      END
```

```
C
C     TAKE IN A NEW V OR U OR BOTH DEPENDING ON IND
C
      IF(IC-NTV-NOM)120,150,150
  120 IF(IND-1)130,130,135
  130 I=I+1
      IF(I-1-NTV)132,131,131
  131 TVI=1.E+07*TSLUT
      GO TO 134
  132 DO 133 J=1,NV
  133 VN(J)=V(J,I)
      TVI=TV(I)
  134 IF(IND.EQ.0) GO TO 20
  135 NOCKE=NO(K)
      UN(NOCKE)=U(NUU+K)
      K=K+1
C
      GO TO 20
C
  150 RETURN
      END
```

```
      SUBROUTINE FUNC(MR,X,FU)
C
C     FUNC IS CALLED BY POWEL TO CALCULATE THE FUNCTION TO BE MINIMIZED. WILL
C     GIVE VALUES ACCORDING TO  F + CSTR*BIVIL.(SEE F AND  BIVIL)
C     PUNISHMENT ON T IS ADDED
C
C     MR- DIMENSION OF X
C     X - THE FREE VARIABLES
C     FU- FUNCTION VALUE
C     COMMON AREA:  SEE SUB1
C
C     REMARK     NOM MUST NOT BE  ZERO
C
C     SUBROUTINES REQUIRED    SRT
C                             SUB1
C                             F
C                             BIVIL
C
      DIMENSION X(1),NO(10),U(20),T(11)
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
C
      DO 10 I=1,NOM
      NO(I)=NOD(I)
      U(I)=X(I)
   10 T(I) = X  (I+NOM)
      CALL SRT(U,NO,T,NOM)
      T(NOM+1)=TSLUT
      DO 20 I=1,NOM
      K=NOM-I+1
   20 U(K+NUU)=U(K)
   40 CALL SUB1(U,NO,T)
      FU = F(U,NO,T)
      FU = FU + CSTR*BIVIL(U,NO,T)
      DO 50 I=1,NOM
   50 FU=FU + 100.*CSTR*(DIM(T(I),TSLUT)**2 + DIM(1.E-07*TSLUT,T(I))**2)
      RETURN
      END
```

```
      SUBROUTINE FUNC(MR,X,FU)
C
C     FUNC IS CALLED BY POWEL TO CALCULATE THE FUNCTION TO BE MINIMIZED. WILL
C     GIVE VALUES ACCORDING TO  F + CSTR*BIVIL.(SEE F AND  BIVIL)
C     PUNISHMENT ON T IS ADDED
C
C     MR- DIMENSION OF X
C     X - THE FREE VARIABLES
C     FU- FUNCTION VALUE
C     COMMON AREA:  SEE SUB1
C
C     REMARK     NOM MUST NOT BE  ZERO
C
C     SUBROUTINES REQUIRED    SRT
C                             SUB1
C                             F
C                             BIVIL
C
      DIMENSION X(1),NO(10),U(20),T(11)
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
C
      DO 10 I=1,NOM
      NO(I)=NOD(I)
      U(I)=X(I)
   10 T(I) = X (I+NOM)
      CALL SRT(U,NO,T,NOM)
      T(NOM+1)=TSLUT
      DO 20 I=1,NOM
      K=NOM-I+1
   20 U(K+NUU)=U(K)
   40 CALL SUB1(U,NO,T)
      FU = F(U,NO,T)
      FU = FU + CSTR*BIVIL(U,NO,T)
      DO 50 I=1,NOM
   50 FU=FU + 100.*CSTR*(DIM(T(I),TSLUT)**2 + DIM(1.E-07*TSLUT,T(I))**2)
      RETURN
      END
```

```
      SUBROUTINE SRT(U,NO,T,N)
C
C
C     SORTS T IN AN INCREASING ORDER. U AND NO ARE CHANGED IN THE SAME
C     WAY AS T.
C
C
C     U -VECTOR SORTED AFTER T
C     NO-VECTOR SORTED AFTER T
C     T -VECTOR SORTED IN AN INCREASING ORDER
C     N -DIMENSIONS OF U,NO AND T
C
C
C     REMARK              N MUST BEAT LEAST ONE
C
C
C      SUBROUTINES REQUIRED            NONE
C
C
      DIMENSION U(N),NO(N),T(N)
C
      N1=N-1
      IF(N1)30,30,7
    7 DO 20 I=1,N1
      NJ=I+1
      DO 20 J=NJ,N
      IF(T(I)-T(J))20,20,10
   10 SL=T(I)
      T(I)=T(J)
      T(J)=SL
      ISL=NO(I)
      NO(I)=NO(J)
      NO(J)=ISL
      SL=U(I)
      U(I)=U(J)
      U(J)=SL
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SRT(U,NO,T,N)
C
C
C     SORTS T IN AN INCREASING ORDER. U AND NO ARE CHANGED IN THE SAME
C     WAY AS T.
C
C
C     U -VECTOR SORTED AFTER T
C     NO-VECTOR SORTED AFTER T
C     T -VECTOR SORTED IN AN INCREASING ORDER
C     N -DIMENSIONS OF U,NO AND T
C
C
C     REMARK            N MUST BEAT LEAST ONE
C
C
C      SUBROUTINES REQUIRED          NONE
C
C
      DIMENSION U(N),NO(N),T(N)
C
      N1=N-1
      IF(N1)30,30,7
    7 DO 20 I=1,N1
      NJ=I+1
      DO 20 J=NJ,N
      IF(T(I)-T(J))20,20,10
   10 SL=T(I)
      T(I)=T(J)
      T(J)=SL
      ISL=NO(I)
      NO(I)=NO(J)
      NO(J)=ISL
      SL=U(I)
      U(I)=U(J)
      U(J)=SL
   20 CONTINUE
   30 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SUB1(U,NO,T)
C
C     CALCULATES THE INITIAL U:S FROM THE REST OF THE U:S AND X/ AND XT  FOR
C     THE SYSTEM       DXDT = B*U + C*V
C
C
C     U -U(1).....U(N) ARE RETURNED CONTAINING THE CALCULATED U:S. U(N+1).....
C        U(N+NOM) ARE LOADED WITH TEH GIVEN U:S
C     NO-NO(I) TELLS WHICH U SHOULD CHANGE TO  U(I+N) AT TIME T(I)
C     T -TIMES OF CHANGE IN U IN TIME-ORDER
C
C     IN COMMON (MUST BE ASSIGNED BEFORE CALLING THE SUBROUTINE):
C                  NOM  -NUMBER OF GHANGES IN U
C                  TSLUT-FINAL TIME
C                  B    -B-MATRIX OF  DXDT = B*U + C*V
C                  C    -THE C-MATRIX
C                  V    -V(I,J) IS THE VALUE OF THE I:TH V IN THE INTERVAL
C                         (TV(J-1) , TV(J) )
C                  TV   -  THE TIMES OF CHANGE IN ANY OF THE V:S IN TIME-ORDER
C                  NOTE:TV(NTV) MUST BE GIVEN THE VALUE TSLUT IN DATA-CARD-FORM
C                  NV   -NUMBER OF V-SIGNALS
C                  NTV  -THE DIMENSION OF TV
C                  N    -THE DIMENSION OF X AND U
C                  XT   -FINAL VALUE OF X
C                  X0   -STARTING VALUE OF X(I.E. THE TANK LEVELS)
C                  XMAX -MAX. ALLOWED DEVIATION OF X FROM ZERO
C                  CSTR -COEFFICIENT OF PUNISHMENT ON RESTRICTIONES
C                  NOD  -THE SAME AS NO , BUT NOD MUST NOT BE CHANGED
C                  XTBEG-MAX.ALLOWED DEVIATION OF X(T) FROM XT(COMPONENTWISE)
C                  XBEG -MAX.VOLUMES OF TANKS
C                  NUU  -THE DIMENSION OF U
C
C     REMARKS  NNU SHOULD BE SET TO THE DECLARED DIMENSION OF BS  AND  SL
C
C     BEFORE CALLING SUB1 DECOM(B,N,NNB,EPS,ISING)  MUST BE CALLED, WHERE NNB
C                  SHOULD BE THE DECLARED DIMENSION OF B, EPS=1.E-07  AND ISING
C                  IS RETURNED ZERO IF EVERYTHING WAS  O.K.,OTHERWISE NOT
C
C     NOM AND NV MUST NOT BE ZERO  AND NTV MUST NOT BE ZERO  OR  ONE
C
C
C     SUBROUTINE REQUIRED            NONE
C
C
      DIMENSION U(1),NO(1),T(1),BS(10),SL(10)
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
C     NNU,NNB AND EPS ARE PARAMETERS FOR DECOM AND SOLVB.NNU SHOULD BE
C     EQUAL TO THE DECLARED DIMENSION OF X  AND U.
C
      NNU=10
      NNB=1
C
C     PUT THE INTEGRALS OF THE V:S INTO SL
C
      DO 10 I=1,NV
      SL1=V(I,1)*TV(1)
      DO 15 J=2,NTV
   15 SL1=SL1+V(I,J)*(TV(J)-TV(J-1))
   10 SL(I)=SL1
```

```
      SUBROUTINE SUB1(U,NO,T)
C
C     CALCULATES THE INITIAL U:S FROM THE REST OF THE U:S AND X/ AND XT  FOR
C     THE SYSTEM       DXDT = B*U + C*V
C
C
C     U -U(1).....U(N) ARE RETURNED CONTAINING THE CALCULATED U:S. U(N+1).....
C        U(N+NOM) ARE LOADED WITH TEH GIVEN U:S
C     NO-NO(I) TELLS WHICH U SHOULD CHANGE TO  U(I+N) AT TIME T(I)
C     T -TIMES OF CHANGE IN U IN TIME-ORDER
C
C     IN COMMON (MUST BE ASSIGNED BEFORE CALLING THE SUBROUTINE):
C                  NOM  -NUMBER OF GHANGES IN U
C                  TSLUT-FINAL TIME
C                  B    -B-MATRIX OF  DXDT = B*U + C*V
C                  C     -THE C-MATRIX
C                  V    -V(I,J) IS THE VALUE OF THE I:TH V IN THE INTERVAL
C                           (TV(J-1) , TV(J) )
C                  TV   -  THE TIMES OF CHANGE IN ANY OF THE V:S IN TIME-ORDER
C                  NOTE:TV(NTV) MUST BE GIVEN THE VALUE TSLUT IN DATA-CARD-FORM
C                  NV    -NUMBER OF V-SIGNALS
C                  NTV   -THE DIMENSION OF TV
C                  N     -THE DIMENSION OF X AND U
C                  XT    -FINAL VALUE OF X
C                  XO    -STARTING VALUE OF X(I.E. THE TANK LEVELS)
C                  XMAX -MAX. ALLOWED DEVIATION OF X FROM ZERO
C                  CSTR -COEFFICIENT OF PUNISHMENT ON RESTRICTIONES
C                  NOD   -THE SAME AS NO , BUT NOD MUST NOT BE CHANGED
C                  XTBEG-MAX.ALLOWED DEVIATION OF X(T) FROM XT(COMPONENTWISE)
C                  XBEG -MAX.VOLUMES OF TANKS
C                  NUU  -THE DIMENSION OF U
C
C     REMARKS  NNU SHOULD BE SET TO THE DECLARED DIMENSION OF BS  AND  SL
C
C     BEFORE CALLING SUB1 DECOM(B,N,NNB,EPS,ISING)  MUST BE CALLED, WHERE NNB
C                  SHOULD BE THE DECLARED DIMENSION OF B, EPS=1.E-07  AND ISING
C                  IS RETURNED ZERO IF EVERYTHING WAS  O.K.,OTHERWISE NOT
C
C     NOM AND NV MUST NOT BE ZERO  AND NTV MUST NOT BE ZERO  OR  ONE
C
C
C     SUBROUTINE REQUIRED            NONE
C
C
      DIMENSION U(1),NO(1),T(1),BS(10),SL(10)
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,XO(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
C     NNU,NNB AND EPS ARE PARAMETERS FOR DECOM AND SOLVB.NNU SHOULD BE
C     EQUAL TO THE DECLARED DIMENSION OF X  AND U.
C
      NNU=10
      NNB=1
C
C     PUT THE INTEGRALS OF THE V:S INTO SL
C
      DO 10 I=1,NV
      SL1=V(I,1)*TV(1)
      DO 15 J=2,NTV
   15 SL1=SL1+V(I,J)*(TV(J)-TV(J-1))
   10 SL(I)=SL1
```

```
C
C
C     CALCULATE THE INTEGRALS OF THE U:S BY SOLVING THE EQUATION SYSTEM
C
      DO 30 I=1,N
      SLAS=0.
      DO 20 J=1,NV
   20 SLAS=SLAS+C(I,J)*SL(J)
   30 BS(I)=XT(I)-X0(I)-SLAS
      CALL SOLVB(BS,SL,NUU,NNB,NNU)
C
C     CALCULATE INITIAL U:S BY SUBTRACTING THE INFLUENCE FROM LATER U:S
C
      DO 75 I=1,NUU
      TT=TSLUT
      SL2=0.
      IF(NOM)35,70,35
   35 DO 60 K=1,NOM
      J=NOM-K+1
      IF(NO(J)-I)60,40,60
   40 TJ = AMAX1(T(J),1.E-07*TSLUT)
      IF(TJ.GE.TSLUT) GO TO 60
      SL2=SL2+U(J+NUU)*(TT-TJ)
      TT=TJ
   60 CONTINUE
   70 U(I)=(SL(I)-SL2)/TT
   75 CONTINUE
      RETURN
      END
```

```
C
C
C     CALCULATE THE INTEGRALS OF THE U:S BY SOLVING THE EQUATION SYSTEM
C
      DO 30 I=1,N
      SLAS=0.
      DO 20 J=1,NV
   20 SLAS=SLAS+C(I,J)*SL(J)
   30 BS(I)=XT(I)-X0(I)-SLAS
      CALL SOLVB(BS,SL,NUU,NNB,NNU)
C
C     CALCULATE INITIAL U:S BY SUBTRACTING THE INFLUENCE FROM LATER U:S
C
      DO 75 I=1,NUU
      TT=TSLUT
      SL2=0.
      IF(NOM)35,70,35
   35 DO 60 K=1,NOM
      J=NOM-K+1
      IF(NO(J)-I)60,40,60
   40 TJ = AMAX1(T(J),1.E-07*TSLUT)
      IF(TJ.GE.TSLUT) GO TO 60
      SL2=SL2+U(J+NUU)*(TT-TJ)
      TT=TJ
   60 CONTINUE
   70 U(I)=(SL(I)-SL2)/TT
   75 CONTINUE
      RETURN
      END
```

```
C
C     CALCULATES THE SUM OF ALL   NORM( ABS(X) - XMAX )**2  FOR DIFFERENT TIMES
C      THAT IS WHEN U OR V CHANGES  AND ABS(X) > XMAX  (NORM IN SUM OF SQUARES).
C
C     ALL NOTATIONS:  SEE SUB1
C
C     REMARK      NV MUST NOT BE ZERO  ;  T(NOM+1)=TSLUT MUST BE SET BEFORE CALL
C
C     NO SUBROUTINE REQUIRED
C
C
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      DIMENSION U(1),NO(1),T(1),UN(20),X(10),VN(10)
C
      IC=0
      STOR=0.
      DO 10 I=1,N
      X(I)=X0(I)
 10   STOR=STOR + DIM(ABS(X(I)),XMAX)**2
      DO 12 I=1,NUU
   12 UN(I)=U(I)
      DO 15 I=1,NV
   15 VN(I)=V(I,1)
C
      T0=0.
      I=1
      K=1
      TVI=TV(1)
C
C     CALCULATE THE NEXT T.IND TELLS IF  U OR V CHANGES THERE
C
   20 IND=2
      IF(TVI-T(K))35,45,55
   35 TT=TVI
      IND=0
      GO TO 70
   45 IND=1
   55 TT=T(K)
      TT=AMAX1(TT,1.E-07*TSLUT)
C
C     CALCULATE THE NEW X AND CHECK
C
   70 DO 100 J=1,N
      SL1=0.
      SL2=0.
      DO 80 NI=1,NUU
   80 SL1=SL1+B(J,NI)*UN(NI)
      DO 90 NI=1,NV
   90 SL2=SL2+C(J,NI)*VN(NI)
  100 X(J)=X(J)+(SL1+SL2)*(TT-T0)
      T0=TT
      IF(IND-1)102,104,102
  102 IFAKT=1
      GO TO 106
  104 IFAKT=2
  106 DO 110 J=1,N
  110 STOR = STOR + FLOAT(IFAKT)*DIM(ABS(X(J)),XMAX)**2
```

```
      FUNCTION F(U,NO,T)

      CALCULATES THE SUM OF ALL   NORM( ABS(X) - XMAX )**2  FOR DIFFERENT TIMES
       THAT IS WHEN U OR V CHANGES  AND ABS(X) > XMAX  (NORM IN SUM OF SQUARES).

      ALL NOTATIONS:  SEE SUB1

      REMARK      NV MUST NOT BE ZERO  ;  T(NOM+1)=TSLUT MUST BE SET BEFORE CALL

      NO SUBROUTINE REQUIRED


      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      DIMENSION U(1),NO(1),T(1),UN(20),X(10),VN(10)
C
      IC=0
      STOR=0.
      DO 10 I=1,N
      X(I)=X0(I)
 10   STOR=STOR + DIM(ABS(X(I)),XMAX)**2
      DO 12 I=1,NUU
   12 UN(I)=U(I)
      DO 15 I=1,NV
   15 VN(I)=V(I,1)
C
      T0=0.
      I=1
      K=1
      TVI=TV(1)

      CALCULATE THE NEXT T.IND TELLS IF  U OR V CHANGES THERE

   20 IND=2
      IF(TVI-T(K))35,45,55
   35 TT=TVI
      IND=0
      GO TO 70
   45 IND=1
   55 TT=T(K)
      TT=AMAX1(TT,1.E-07*TSLUT)
C
C     CALCULATE THE NEW X AND CHECK
C
   70 DO 100 J=1,N
      SL1=0.
      SL2=0.
      DO 80 NI=1,NUU
   80 SL1=SL1+B(J,NI)*UN(NI)
      DO 90 NI=1,NV
   90 SL2=SL2+C(J,NI)*VN(NI)
  100 X(J)=X(J)+(SL1+SL2)*(TT-T0)
      T0=TT
      IF(IND-1)102,104,102
  102 IFAKT=1
      GO TO 106
  104 IFAKT=2
  106 DO 110 J=1,N
  110 STOR = STOR + FLOAT(IFAKT)*DIM(ABS(X(J)),XMAX)**2
```

```
      IC=IC+IFAKT
      IF(I-NTV)116,112,116
 112  IF(IND-1)113,113,116
 113  DO 114 J=1,N
 114  STOR=STOR+DIM(ABS(X(J)-XT(J)),XTBEG(J))**2
 116  CONTINUE
C
C     TAKE IN A NEW V OR U OR BOTH DEPENDING ON IND
C
      IF(IC-NTV-NOM)120,150,150
  120 IF(IND-1)130,130,135
  130 I=I+1
      IF(I-1-NTV)132,131,131
  131 TVI=1.E+07*TSLUT
      GO TO 134
  132 DO 133 J=1,NV
  133 VN(J)=V(J,I)
      TVI=TV(I)
  134 IF(IND.EQ.0) GO TO 20
  135 NOCKE=NO(K)
      UN(NOCKE)=U(NUU+K)
      K=K+1
C
      GO TO 20
C
  150 F = STOR/N/(NOM+NTV)
      RETURN
      END
```

```
      IC=IC+IFAKT
      IF(I-NTV)116,112,116
  112 IF(IND-1)113,113,116
  113 DO 114 J=1,N
  114 STOR=STOR+DIM(ABS(X(J)-XT(J)),XTBEG(J))**2
  116 CONTINUE
C
C     TAKE IN A NEW V OR U OR BOTH DEPENDING ON IND
C
      IF(IC-NTV-NOM)120,150,150
  120 IF(IND-1)130,130,135
  130 I=I+1
      IF(I-1-NTV)132,131,131
  131 TVI=1.E+07*TSLUT
      GO TO 134
  132 DO 133 J=1,NV
  133 VN(J)=V(J,I)
      TVI=TV(I)
  134 IF(IND.EQ.0) GO TO 20
  135 NOCKE=NO(K)
      UN(NOCKE)=U(NUU+K)
      K=K+1
C
      GO TO 20
C
  150 F = STOR/N/(NOM+NTV)
      RETURN
      END
```

```
      FUNCTION BIVIL(U,NO,T)
C
C     CALCULATES THE DEVIATION FROM THE ALLOWED AREA IN U  AND  S = D*U + E*V AT
C     ALL TIMES  AND  IS GIVEN A NUMBER RELATED TO THE DEVIATION
C      ( (SUM OF ALL DEVIATIONS)**2 )
C
C     U, NO, T - SEE SUB1
C     COMMON- SEE SUB1
C
C     IN COMMON /BIVI/ (MUST BE ASSIGNED VALUES BEFORE CALLING BIVIL) :
C                    D    - D-MATRIX IN  S = D*U + E*V
C                    E    - E-MATRIX
C                    UMIN-AT UMIN(I)  WE START PUNISH  U(I)
C                    UMAX-CORRESPONDING TO UMIN
C                    SMAX AND SMIN-SAME AS UMIN AND UMAX BUT  FOR  S
C
C     REMARK    NV MUST NOT BE ZERO   ; T(NOM+1)=TSLUT MUST BE SET BEFORE CALL
C
C     NO SUBROUTINE REQUIRED
C
      DIMENSION UN(20),VN(10),U(1),NO(1),T(1)
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      COMMON /BIVI/ D(10),E(10),UMIN(10),UMAX(10),SMAX,SMIN
      STOR=0.
      IC=0
C
C     CHECK IF THE U:S ARE WITHIN THE LIMITS
C
      NU=NUU+NOM
      DO 20 I=1,NU
      IF(I-NUU)4,4,6
 4    NOCKE=I
      GO TO 20
 6    NOCKE = NO(I-NUU)
 20   STOR = STOR +      DIM(U(I),       100.)**2+     DIM(UMIN(NOCKE),
     VU(I))**2
C
C CALCULATE AND CHECK S AT ALL TIMES
C
C     TAKE IN STARTING VALUES OF U AND V
C
      DO 50 I=1,NUU
 50   UN(I)=U(I)
      DO 60 I=1,NV
 60   VN(I)=V(I,1)
      TVI=TV(1)
      T0=0.
      I=1
      K=1
C
C     CALCULATE THE NEXT INTERESTING T
C
 70   IND=2
      IF(TVI-T(K))90,100,110
 90   TT=TVI
      IND=0
      GO TO 120
 100  IND=1
 110  TT=T(K)
      TT=AMAX1(TT,1.E-07*TSLUT)
```

```
      FUNCTION BIVIL(U,NO,T)
C
C     CALCULATES THE DEVIATION FROM THE ALLOWED AREA IN U  AND  S = D*U + E*V AT
C     ALL TIMES  AND  IS GIVEN A NUMBER RELATED TO THE DEVIATION
C      ( (SUM OF ALL DEVIATIONS)**2 )
C
C     U, NO, T - SEE SUB1
C     COMMON- SEE SUB1
C
C     IN COMMON /BIVI/ (MUST BE ASSIGNED VALUES BEFORE CALLING BIVIL) :
C                      D   - D-MATRIX IN  S = D*U + E*V
C                      E   - E-MATRIX
C                      UMIN-AT UMIN(I)  WE START PUNISH  U(I)
C                      UMAX-CORRESPONDING TO UMIN
C                      SMAX AND SMIN-SAME AS UMIN AND UMAX BUT  FOR  S
C
C     REMARK    NV MUST NOT BE ZERO   ; T(NOM+1)=TSLUT MUST BE SET BEFORE CALL
C
C     NO SUBROUTINE REQUIRED
C
      DIMENSION UN(20),VN(10),U(1),NO(1),T(1)
      COMMON NOM,TSLUT,B(10,10),C(10,10),V(10,10),TV(10),NV,NTV,N,XT(10)
     B,X0(10),XMAX,CSTR,NOD(10),XTBEG(10),XBEG(10),NUU
      COMMON /BIVI/ D(10),E(10),UMIN(10),UMAX(10),SMAX,SMIN
      STOR=0.
      IC=0
C
C     CHECK IF THE U:S ARE WITHIN THE LIMITS
C
      NU=NUU+NOM
      DO 20 I=1,NU
      IF(I-NUU)4,4,6
 4    NOCKE=I
      GO TO 20
 6    NOCKE = NO(I-NUU)
 20   STOR = STOR +      DIM(U(I),        100.)**2+     DIM(UMIN(NOCKE),
     VU(I))**2
C
C CALCULATE AND CHECK S AT ALL TIMES
C
C     TAKE IN STARTING VALUES OF U AND V
C
      DO 50 I=1,NUU
 50   UN(I)=U(I)
      DO 60 I=1,NV
 60   VN(I)=V(I,1)
      TVI=TV(1)
      T0=0.
      I=1
      K=1
C
C     CALCULATE THE NEXT INTERESTING T
C
 70   IND=2
      IF(TVI-T(K))90,100,110
 90   TT=TVI
      IND=0
      GO TO 120
 100  IND=1
 110  TT=T(K)
      TT=AMAX1(TT,1.E-07*TSLUT)
```

```
C
C     CHECK OF THE LAST S
C
 120  S=0.
      DO 130 J=1,NUU
 130  S=S+D(J)*UN(J)
      DO 140 J=1,NV
 140  S=S+E(J)*VN(J)
      IF(IND-1)142,144,142
 142  IFAKT=1
      GO TO 146
 144  IFAKT =2
 146  STOR=STOR +((TT-T0)*(DIM(S,100.)+DIM(SMIN,S)))**2
      IC=IC+IFAKT
      T0=TT
C
C     NEW V:S OR U:S ARE TAKEN IN
C
      IF(IC-NTV-NOM)160,230,230
 160  IF(IND-1)170,170,220
 170  I=I+1
      IF(I-1-NTV)200,180,180
 180  TVI=1.E+07*TSLUT
      GO TO 215
 200  DO 210 J=1,NV
 210  VN(J)=V(J,I)
      TVI=TV(I)
 215  IF(IND.EQ.0) GO TO 70
 220  NOCKE=NO(K)
      UN(NOCKE)=U(NUU+K)
      K=K+1
C
      GO TO 70
C
 230  BIVIL = STOR/(NU+NOM+NTV)
      RETURN
      END
```

```
C
C      CHECK OF THE LAST S
C
 120   S=0.
       DO 130 J=1,NUU
 130   S=S+D(J)*UN(J)
       DO 140 J=1,NV
 140   S=S+E(J)*VN(J)
       IF(IND-1)142,144,142
  142  IFAKT=1
       GO TO 146
  144  IFAKT =2
  146  STOR=STOR +((TT-T0)*(DIM(S,100.)+DIM(SMIN,S)))**2
       IC=IC+IFAKT
       T0=TT
C
C      NEW V:S OR U:S ARE TAKEN IN
C
       IF(IC-NTV-NOM)160,230,230
  160  IF(IND-1)170,170,220
  170  I=I+1
       IF(I-1-NTV)200,180,180
  180  TVI=1.E+07*TSLUT
       GO TO 215
 200   DO 210 J=1,NV
 210   VN(J)=V(J,I)
       TVI=TV(I)
  215  IF(IND.EQ.0) GO TO 70
 220   NOCKE=NO(K)
       UN(NOCKE)=U(NUU+K)
       K=K+1
C
       GO TO 70
C
  230  BIVIL = STOR/(NU+NOM+NTV)
       RETURN
       END
```