A Program for the Interactive Solution of Parametric Optimization Problems in Dynamic Systems

Glad, Torkel

1974

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

A PROGRAM FOR THE INTERACTIVE SOLUTION OF PARAMETRIC OPTIMIZATION PROBLEMS IN DYNAMIC SYSTEMS

TORKEL GLAD

Report 7424 November 1974
Lund Institute of Technology
Division of Automatic Control

A PROGRAM FOR THE INTERACTIVE SOLUTION OF PARAMETRIC OPTIMIZATION PROBLEMS IN DYNAMIC SYSTEMS[†]

T. Glad

ABSTRACT

A method for optimizing dynamic systems with respect to parameters has been included in an interactive simulation program. There is no restriction to a special class of systems or criteria. The numerical method is based on a Quasi-Newton method, which uses a combination of Lagrange multipliers and a penalty function to handle constraints.

[†]This work has been supported by the Swedish Institute of Applied Mathematics.

TABLE OF CONTENTS

Introduction.

The problem of designing control systems.

The general control problem is described in fig. 1.

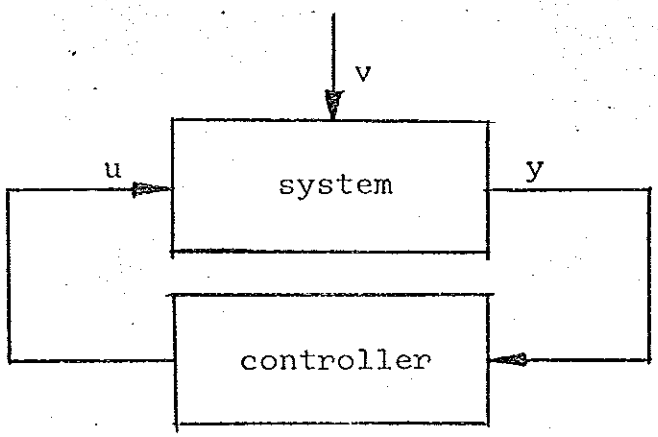


Fig. 1

The system can be influenced by an input u, and information about it is given by the output y. The system is also influenced by disturbances v coming from its environment. The problem then consists of finding a controller which acts on the information y to produce a control u in such a way that the behaviour of the system is "good" despite the disturbances v.

Typically the system could be described by differential equations

$$\dot{x}(t) = f(x(t), u(t), v(t), t)$$

$$y(t) = g(x(t), u(t), v(t), t)$$

The disturbance v might be a stochastic process or a time function of a certain type, e.g. a step function or a ramp.

Classical design techniques are often limited to linear systems and many of them are not systematic i.e. they require the designer to make arbitrary choices at some stage of the design process. A powerful way of making the design procedure systematic is to formulate a mathematical criterion which measures how good the system is. The input u should then be chosen to minimize this criterion. A typical optimization problem could be of the form

$$\text{minimize} \quad J = \int_0^T L(x,u,t)dt + F(x(T))$$

where x is given by

$$\dot{x} = f(x,u,t) \qquad x(0) = x_0$$

under the constraints

$$\psi(x(T)) = 0$$

$$g(u) \leq 0$$

There are several problems involved in this approach. Since an analytic solution can only be found in special cases, one must rely on numerical techniques. There are several algorithms for solving the above problem numerically. These algorithms, however, give as a solution the open-loop control program u(t) for a given value of x_0. What is usually desired to find is a feedback solution u = k(x) which gives the optimum value of J for all starting points. This can only be done in

special cases. Two of these are linear systems with a quadratic loss function and no constraints, and the minimum time problem with bounded input which gives a "bang-bang" solution.

Even when the feedback law k(x) can be found there might be problems. The function k(x) can be very complicated making it difficult to implement it (this is often the case for bang-bang problems). Furthermore the feedback law u = k(x) usually depends on all components of the state vector x. If these are not directly measurable (i.e. y = x), they have to be reconstructed from y. For a linear system this can be done using a Kalman filter, but if there are many states it means that the controller will be quite complicated.

A different approach, which avoids these disadvantages, is to specify a control law u = k(y,p) of a given structure. All freedom within this structure lies in the parameter vector p. This vector has to be chosen to minimize the given criterion, which results in a finite dimensional optimization problem possibly with constraints. In the following chapters a program which can solve this problem interactively is described.

The optimization program as a part of a simulation program.

To solve the optimization problem it is necessary or desirable to do things that are normally done by simulation programs: solve a system of differential and difference equations and plot the results for different values of the parameters. It is then natural to try to introduce the optimization method into an already existing simulation program. There is at the Division of Automatic Control a

program package called SIMNON for interactive simulation of nonlinear systems, [1]. In SIMNON a system consisting of interconnected continous time and discrete time subsystems can be simulated interactively and the result plotted on a display. The subsystems are described in a special language and can easily be changed from one simulation to an other. It is also possible to include subsystems described in FORTRAN. The connections between different subsystems are described in a special connecting system.

To perform the optimization a starting value of the parameter vector p, which is to be varied in the optimization, is given. The system is then simulated for this value, and the criterion and the constraints are evaluated. The optimization routine then has to use this information to decide on a new value of the parameter vector and the process is repeated. The situation can be described by fig. 2.

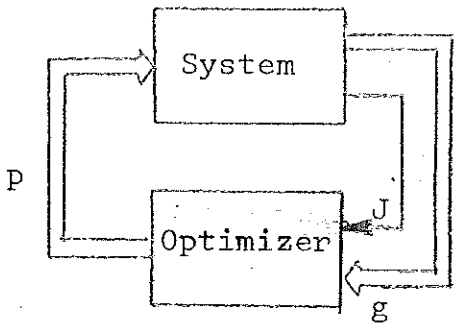


Fig. 2

Then criterion J and the constraint vector g are connected from the system to the optimizer while the output of the

optimizer is the vector p, which is an input to the system. The system can consist of a number of subsystems as shown in figure 3. The connections between these subsystems, as well as connections needed to form J and g from outputs of the subsystems, are made in the connecting system. At each sampling point of the optimizer it uses the values of J and g to compute the next value of p. If the criterion or constraints depend explicitly on the parameter vector p, a vector p_d, containing delayed values of p, is used.

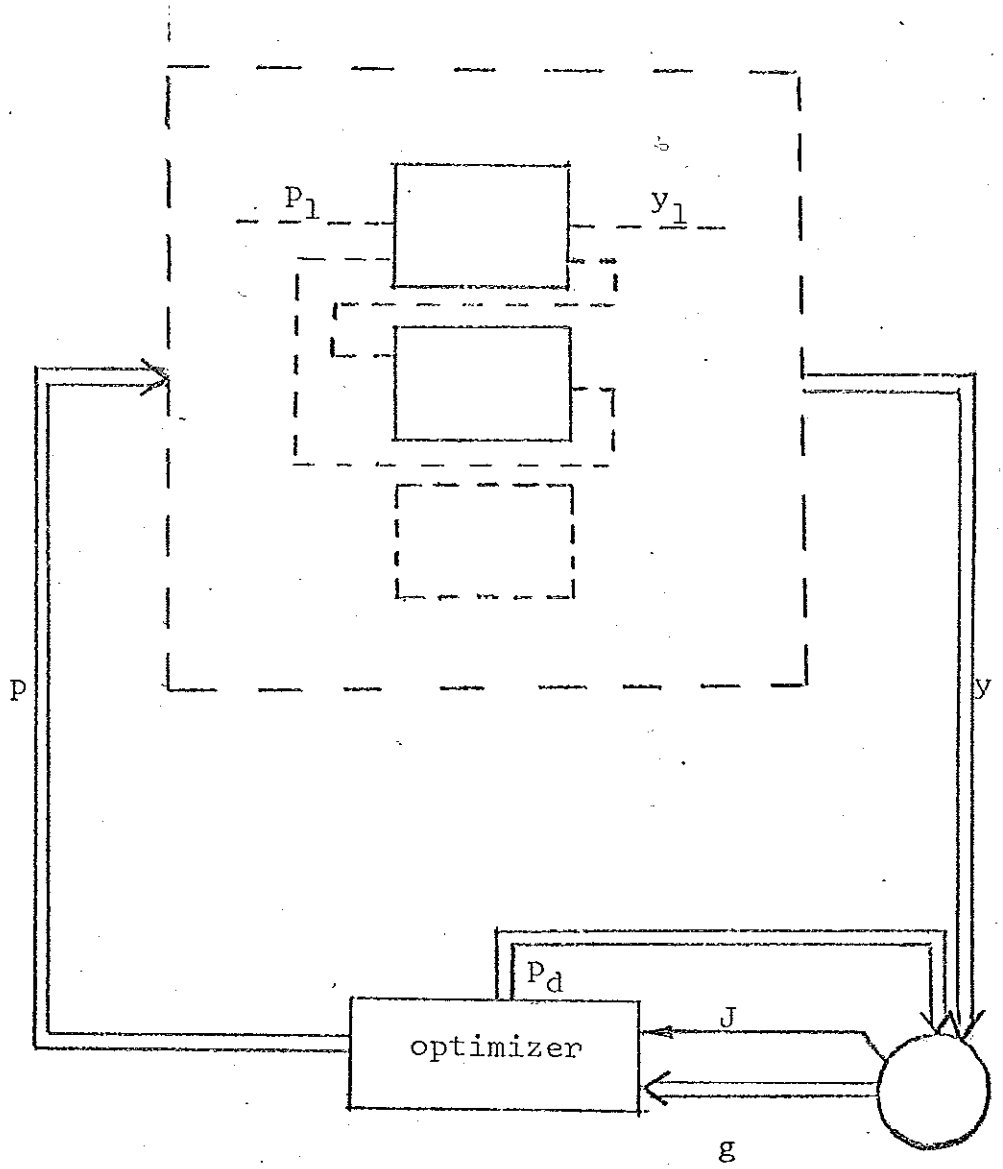


Fig. 3

The optimization routine.

From the previous reasoning it follows that this routine should be able to perform the minimization of a function under equality and inequality constraints. The routine should be based on the values of the criterion and the constraints and not require their derivatives to be calculated explicitly. In many cases it is possible to include the computation of derivatives in the system where J and g_i are computed. Since it means much more work for the user when writing down the systems, it has however been avoided.

Let the criterion be J(p) and the constraints $g_i(p) = 0$ $i = 1,\ldots,k$ and $g_i(p) \leq 0$, $i = k+1,\ldots,m$. The algorithm can then be described as follows.

The function

$$F(p,\lambda) = J(p) + \frac{1}{c} \sum_{i=1}^{k} \{(cg_i(p) + \frac{1}{2}\lambda_i)^2 - \frac{1}{4}\lambda_i^2\} +$$

$$+ \frac{1}{c} \sum_{i=k+1}^{m} \{(cg_i(p) + \frac{1}{2}\lambda_i)_+^2 - \frac{1}{4}\lambda_i^2\}$$

is formed†.

This function is minimized for a fixed value of λ using an unconstrained minimization routine. Then λ is updated according to the formula

$\lambda_i := \lambda_i + 2cg_i$ if $i = 1,\ldots,k$ and

$\lambda_i := \max(\lambda_i+2cg_i, 0)$ if $i = k+1,\ldots m$

---

† $(cg_i+\lambda_i)_+ = \begin{cases} cg_i + \lambda_i & \text{if } cg_i + \lambda_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$

The theory behind this algorithm can be found in [2], [3], and only a short account is given here.

It is assumed that the optimum, p^*, is a Kuhn-Tucker point, i.e. there exists a vector $\lambda^* = (\lambda_1^*,\ldots,\lambda_m^*)^T$ such that

$$L_p(p^*,\lambda^*) = 0$$

$$g_i(p^*) = 0 \qquad i = 1,\ldots,k$$

$$g_i(p^*) \leq 0 \qquad i = k+1,\ldots,m$$

$$\lambda_i \geq 0 \qquad i = k+1,\ldots,m$$

$$\lambda_i\, g_i(p^*) = 0 \qquad i = k+1,\ldots,m$$

where

$$L(p,\lambda) = J(p) + \sum_{i=1}^{m} \lambda_i g_i(p)$$

It is also assumed that the optimum is nonsingular, i.e. for all vectors y such that

$$(g_i)_p(p^*)y = 0 \qquad i = 1,\ldots,k$$

and

$$(g_i)_p(p^*)y = 0 \quad \text{all } i\in[k+1,\ldots,m] \quad \text{with } g_i(p^*) = 0$$

it follows that

$$y^T L_{pp}(p^*,\lambda^*)y > 0.$$

This means that L_{pp} is positive definite in all directions which lie along the active constraints. To ensure that the second derivative shown below is continuous at p^* it is also

assumed that $\lambda_i > 0$ for all $i\in[k+1,\ldots,m]$ with $g_i(p^*)=0$ (this condition is called strict complementarity). It then follows that

$$F_p(p^*,\lambda^*) = J_p(p^*) + 2\sum_{i=1}^{k}(cg_i(p^*) + \tfrac{1}{2}\lambda_i^*)(g_i)_p(p^*) +$$

$$+ 2\sum_{i=k+1}^{m}(cg_i(p^*) + \tfrac{1}{2}\lambda_i^*)_+(g_i)_p(p^*) = L_p(p^*,\lambda^*) = 0$$

$$F_{pp}(p^*,\lambda^*) = L_{pp}(p^*,\lambda^*) + 2\sum_{i=1}^{k} c(g_i)_p{}^T(g_i)_p +$$

$$+ 2\sum_{\substack{i=k+1\\ g_i=0}}^{m} c(g_i)_p{}^T(g_i)_p$$

Since it is possible to prove that $F_{pp}(p^*,\lambda^*)$ is positive definite if c is chosen large enough, it follows that $F(p,\lambda^*)$ has an unconstrained local minimum at p^*. The updating formula for λ will ensure that $\lambda \to \lambda^*$ if c is large enough. The user can give the initial value of c, which can be increased by the algorithm if the convergence is too slow. In this way convergence can be obtained even if some of the conditions on the minimum stated above are not true. (The convergence might be slow in this case.)

The unconstrained minimization of the function $F(p,\lambda)$ is performed by the subroutine NUFLE1, which is a Quasi-Newton algorithm using difference approximations of the derivatives. It is based on the algorithm VA10A, [4]. At the start of each iteration the algorithm has available a point p, a matrix B, which approximates the second derivative, and a vector z, which contains the difference approximations of the gradient. A search direction s is then calculated from $Bs = -z$ and an

approximate minimum along the line ($p+\alpha s$, $\alpha > 0$) gives the new value of p. A new vector z is computed from difference approximations and B is updated in such a way that $\hat{B}\,\Delta p = \Delta z$, where $\hat{B}$ is the new value of B.

In the original version of NUFLE1 the calculation of the function value is done by a subroutine which is called for every new value of p. This means that the optimization algorithm controls the calculation of the function. In order to use the algorithm in the simulation program SIMNON one must do the opposite. Each time the system has been simulated and the criterion computed the optimization program must be called to produce the next value of the parameter vector p. In order to change as little as possible in the subroutine the following modification has been made. Each call statement to the subroutine which computes function values is replaced by a RETURN. An integer variable is set to indicate from where the return was made, making it possible to continue the minimization at the right point in NUFLE1.

The outer loop of the minimization, i.e. the updating of λ, is done in a subroutine OPTA, which is written as a discrete time FORTRAN system in SIMNON. The basic structure of OPTA is given in Fig. 4.

There are four different entry points A, B, C and D. When the command SYST is given, OPTA is entered at A where the inputs, outputs, variables, etc., which have to be available for SIMNON, are defined. OPTA is also entered at B where the parameters of the optimization routine are given initial values. These parameters are listed in table 1, together with the values they are given. The user can change these values with the command PAR. When the command SIMU is given, OPTA is entered at C where the minimization is initialized. During the optimization OPTA is

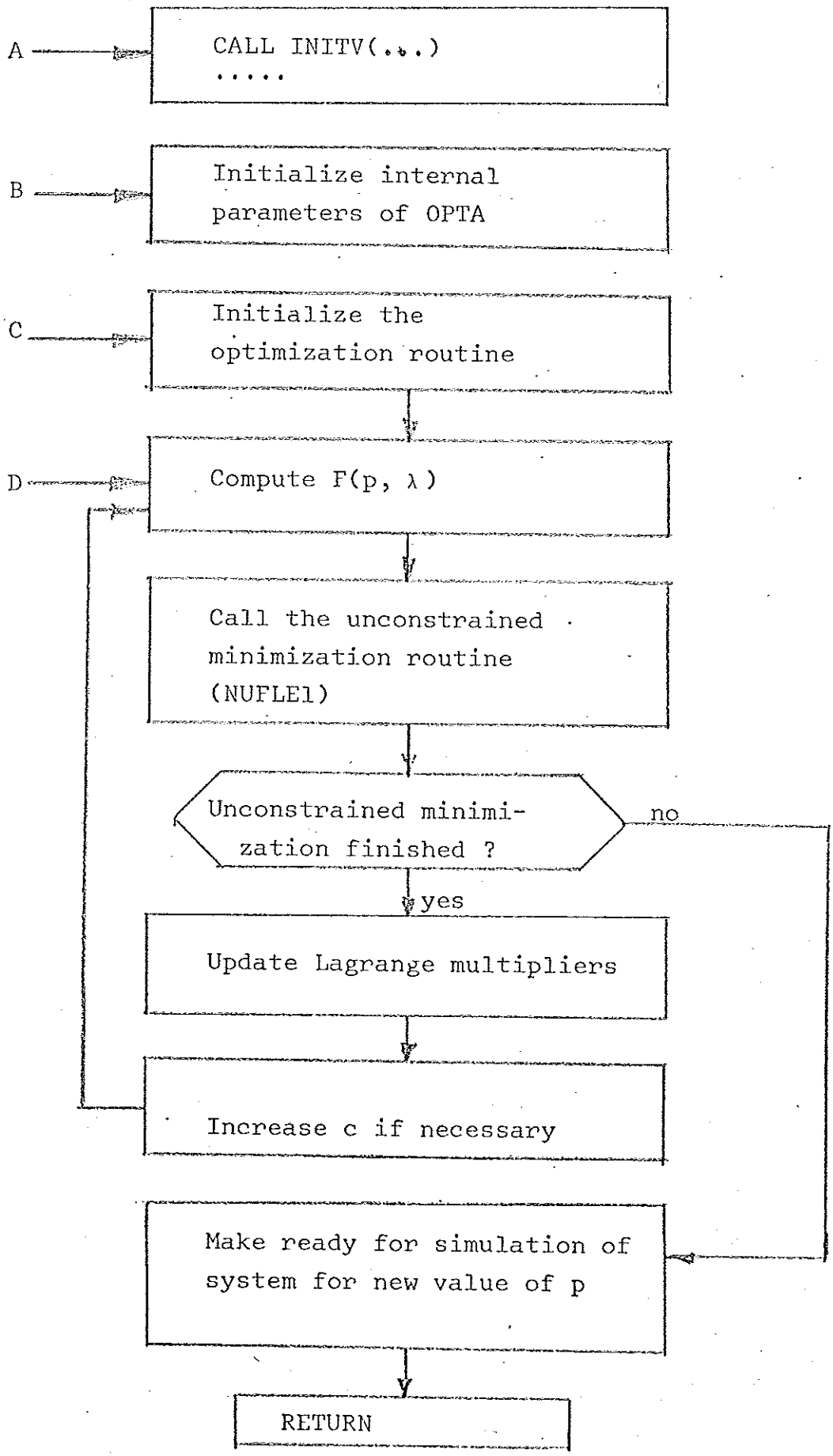

A
CALL INITV(...)
.....
B
Initialize internal parameters of OPTA
C
Initialize the optimization routine
D
Compute F(p, λ)
Call the unconstrained minimization routine (NUFLE1)
Unconstrained minimization finished ?
no
yes
Update Lagrange multipliers
Increase c if necessary
Make ready for simulation of system for new value of p
RETURN

Fig. 4

entered at D at each sampling point. To start with, p is given the values PI1,...,PIn supplied by the user. The system is simulated with these values of the parameters for a time interval of length TINC and OPTA is then entered. The inputs LOSS[OPTA] and CONi[OPTA] now contain the values of the criterion and constraints. These values are used in NUFLE1 to calculate a new value of p which forms the output vector P1,...Pn of OPTA. The states of all the simulated systems are reset and the procedure is repeated.

When NUFLE1 has found an unconstrained minimum, the Lagrange multipliers λ are adjusted and a new unconstrained minimization is started. Constraint satisfaction is judged by the test quantity TEST which is defined as

$$TEST = \max(|g_1|,\ldots,|g_k|,|\min(\lambda_{k+1},-g_{k+1})|,\ldots,|\min(\lambda_m,-g_m)|)$$

If TEST = 0 at a point where $F(x,\lambda)$ has a local minimum, it follows from the Kuhn-Tucker conditions that this point is a local minimum to the constrained problem. The algorithm is stopped when TEST is less than the parameter DELTA which can be set by the user. If TEST does not decrease fast enough, the parameter c is increased. During the first part of the minimization NUFLE1 calculates the gradients by taking forward differences of the function values. When the minimum is almost reached, it starts using central differences. This is shown by the variable IDIFF, which is 1 for forward differences and 2 for central differences.

How to use the program.

The system to be optimized is given as a collection of SIMNON systems as described in [1]. The parameters which

are to be varied by the optimization routine should be declared as inputs and the information needed to calculate the constraints and criterion as outputs or states. In the connecting system they are tied together with the optimization routine OPTA. The criterion J is the input LOSS of OPTA and the constraint vector g corresponds to the inputs CON1,CON2,.. of OPTA. Observe that the sign of J should be chosen for J to be minimized. The constraints must be of the form $g_1=0, g_2=0, .., g_k=0, g_{k+1} \leq 0, .., g_m \leq 0$, i.e. the equality constraints are placed first. The connecting system will then have the structure of fig.5 (y1,y2,.. are outputs or states of the systems syst1,syst2,...).

```
CONNECTING SYSTEM CONN

Internal connections

LOSS[OPTA] =
CON1[OPTA] =   } functions of PD1[OPTA],..,PDn[OPTA],
               } y1[syst1],y2[syst1],..,y1[syst2],..
CONm[OPTA] =

p1[syst1] = P1[OPTA]
p2[syst1] = P2[OPTA]

p1[syst2] = Pi[OPTA]

END
```

fig. 5

PD1,PD2,.. are delayed values of P1,P2,.. and are used instead of P1,P2,.. when LOSS or CON1,CON2,.. depend explicitly on p. In fig. 5 it is assumed that y1,y2,.. depend on p1,p2,.. only via the differential or difference equations and not explicitly. If an output of a system depends directly on a component of P then the corresponding component of PD has to be introduced into the system.

The number of parameters in p and the number of constraints are set by the commands
SET NVAR:n and SET NCONS:m
Then the command
SYST syst1 syst2 ... OPTA CONN
is given.
When this is done the user can set the initial values of the components of p, which lie in the vector PI, with the command INIT. The internal parameters of OPTA can be set using the command PAR. These parameters are listed in table 1 together with their default values. The response of the system can be plotted in the usual way during optimization. If some variables are to be plotted against time, a straightforward plot would give all the curves for different values of the parameter vector after each other (fig.6). Often it is more convenient to plot them all from the same starting point as shown in fig. 7 so that they can be compared directly.

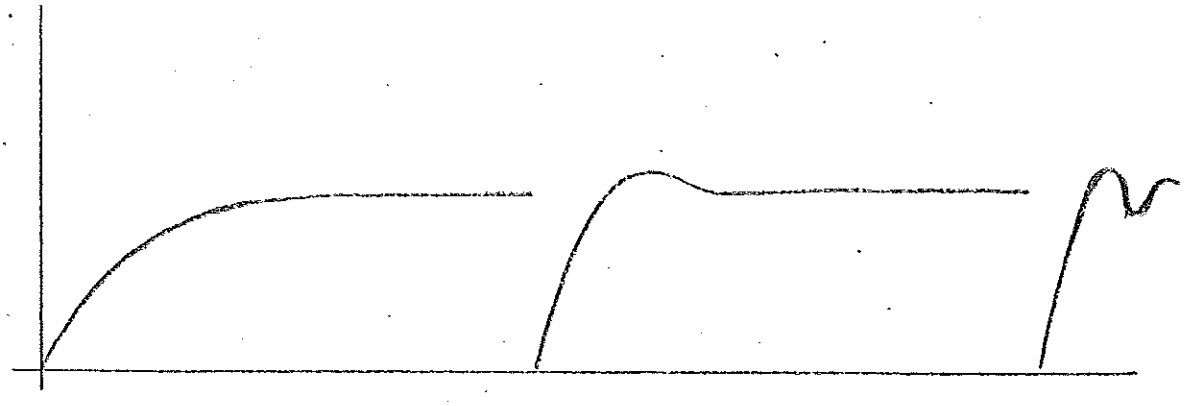


Fig. 6

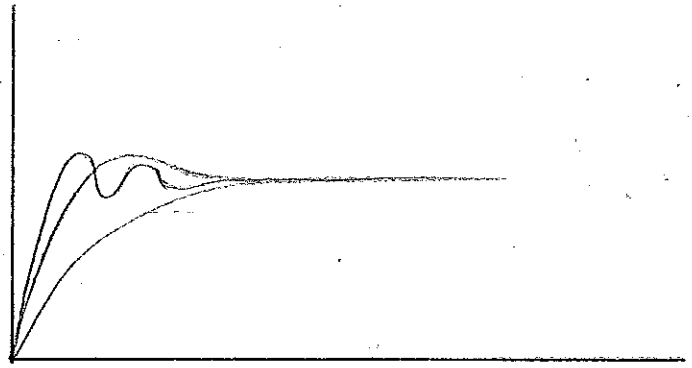


Fig. 7

The easiest way to accomplish this is to make the plot against an extra state variable T, with the differential equation DT = 1. Since T will be reset together with the other states, the desired result is obtained. Finally, the command SIMU will start the optimization which will continue until either the minimization is finished or the time interval specified in SIMU has elapsed.

The results of the minimization are printed on line printer (if PRIN ≠ 0). Also the minimization can be interrupted at any point with the data switches 0 and 1. All internal variables, parameters, states etc. can then be displayed using the command DISP. The meaning of the variables in OPTA which are displayed is given in table 2. The optimization can be restarted with SIMU-CONT.

For a detailed description of available commands in SIMNON see [1].

Table 1

Internal parameters in OPTA
(numbers within parantheses are default values)

XM1 XM2 ...	(1 1 ..)	Scaling factors, see HH and EPS
LAM1 LAM2 ..	(0 0 ..)	Lagrange multipliers, initial values
DFN	(-0.5)	Controls initial step at the first linear minimization; should give an estimate of the likely reduction in function value, Δf; there are two possibilities: DFN>0 DFN itself is an estimate of Δf DFN<0 ABS(DFN)•f is taken as an estimate of Δf
TINC	(1)	Length of sampling interval of OPTA
HH	(0.005)	Step length for calculation of the gradient by differences; the step length for each component Pi is XMi•HH
EPS	(0.01)	Stopping criterion for unconstrained minimization - is satisfied when the change in each component Pi is less than XMi•EPS
PRIN	(1)	Controls printout on line printer; every ABS(PRIN):th iteration is printed; if PRIN<0 only function values are printed; if PRIN>0 also P and the gradient are printed; if PRIN=0 there is no printout
EVMAX	(10000)	Maximum number of function evaluations

CEQ (0) Number of equality constraints

C (1) Constant used in the modified function see page 6; only used for constrained problems

DELTA (0.01) Stopping criterion for constrained minimization - satisfied when TEST (see page 10) is less than DELTA

RESET (1) The states are reset to their initial values if RESET>0

DARK (1) There is no trace on the display at the sampling points of OPTA if DARK>0

MODE (1) Controls the initialization of the approximation of the second derivative, H

MODE=1 H is set equal to the identity matrix initially

MODE=3 the H-matrix from the previous minimization is used

Table 2

Explanation of variables etc. in OPTA shown under the command DISP

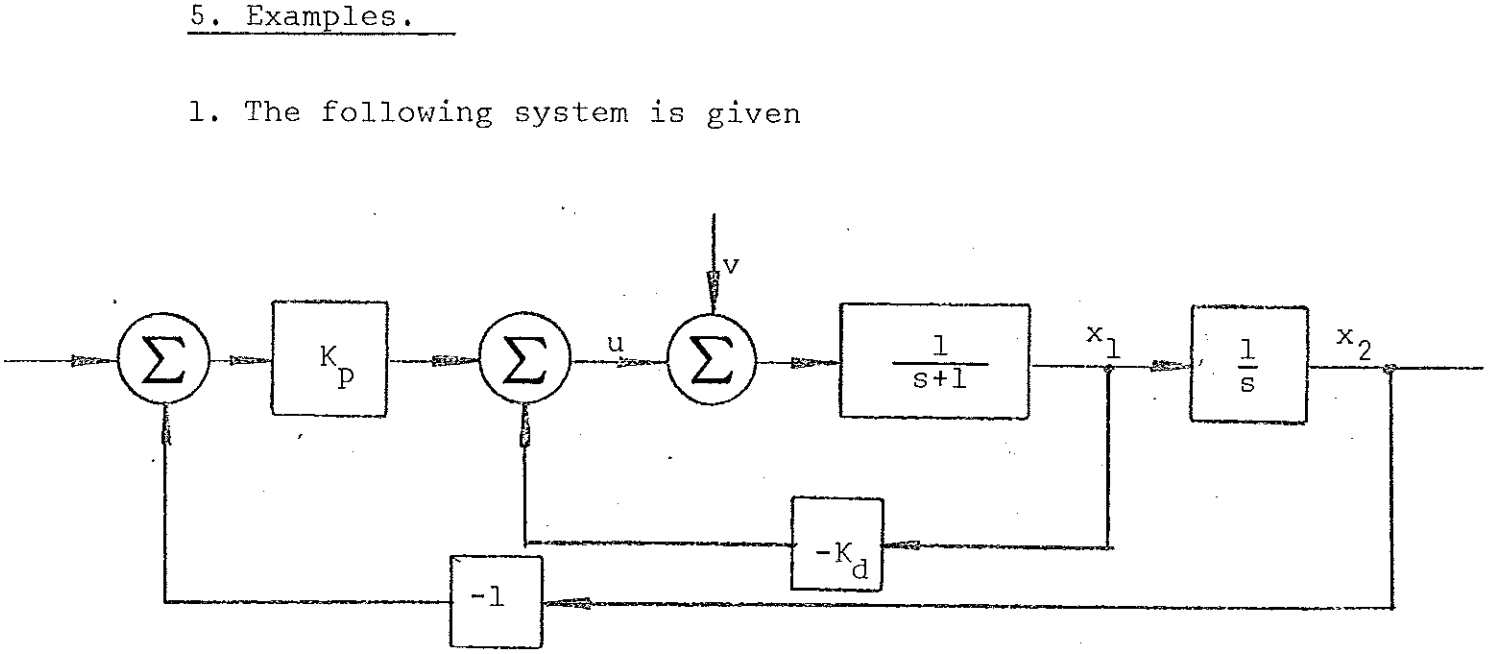
STATE	
PD1 PD2...	Value of parameter vector at previous sampling point
INIT	
PI1 PI2...	Initial value of parameters
NEW	
PDN1 PDN2..	Same as P1 P2...
INPUT	
CON1 CON2..	Present values of the constraints
LOSS	Present value of the criterion to be minimizaed
OUTPUT	
P1 P2...	Present values of the parameters
TSAMP	
TS	Next sampling point of OPTA
PAR	see table 1
VAR	
RI	Number of evaluations of the system

VAR cont'd

GR1 GR2...	Latest calculated gradient
IND	Indicates what part of OPTA is going to be executed next
IDIFF	If IDIFF=1 forward differences are used in the calculation of the gradient while if IDIFF=2 central differences are used
FVAL	Function value at previous sampling point
TEST	Convergence test quantity, see page 10

5. Examples.

1. The following system is given



The purpose of the controller containing the parameters K_p and K_d is to keep $x_2$as small as possible despite the disturbance v. It is assumed that v(t) is an impulse disturbance. This is equivalent to setting $x_1(0)=1$. The criterion is

$$J=\int_0^T x_2^2\,dt$$

It is also assumed that the total control effort is limited.

$$g=\int_0^T u^2\,dt - u_{lim} \leq 0 \qquad\qquad u_{lim}= 0.5$$

Below are given the system IMP and connecting system CONN needed for this problem.

```
CONTINUOUS SYSTEM IMP
STATE X1 X2  Z W T
DER DX1 DX2 DZ DW DT
INPUT KD KP
OUTPUT Y

OUTPUT
Y=X2

DYNAMICS
U=-KD*X1-KP*X2
DX1=-X1+U
DX2=X1
DZ=U*U
DW=X2*X2
DT=1.

END

- - - - - - - - - - - - - - - - - - - - - - - - - - - -

CONNECTING SYSTEM CONN


WT:10
LOSS[OPTA]=WT*W[IMP]
ULIM:.5
CON1[OPTA]=Z[IMP]-ULIM

KD[IMP]=P1[OPTA]
KP[IMP]=P2[OPTA]

END
```

To do the optimization the following commands are given.

```
>SET NPAR:2
>,NCONS:1
>SYST IMP OPTA CONN
>INIT X1:1
>,PI1:2
>,PI2:2
>PAR TINC:10
>,PRIN:5
>PLOT Y CON1 (T)
>AXES H 0 10 V -
>SIMU 0 10000 1
```

At the beginning of the optimization the curves of fig. 8 are displayed. Fig. 9 shows the result at the end of the optimization. In fig. 10 the printout on the line printer is given. Fig. 11 shows the values of states, outputs etc. of the systems at the end of the optimization.

Remark: If $T=\infty$ the problem can be solved analytically and

$$J = \frac{1}{2(K_d+1)K_p} \qquad g = \frac{K_d^2 + K_p}{2(1+K_d)} - 0.5$$

The solution is $K_d = 0.8165$ $K_p = 1.1498$

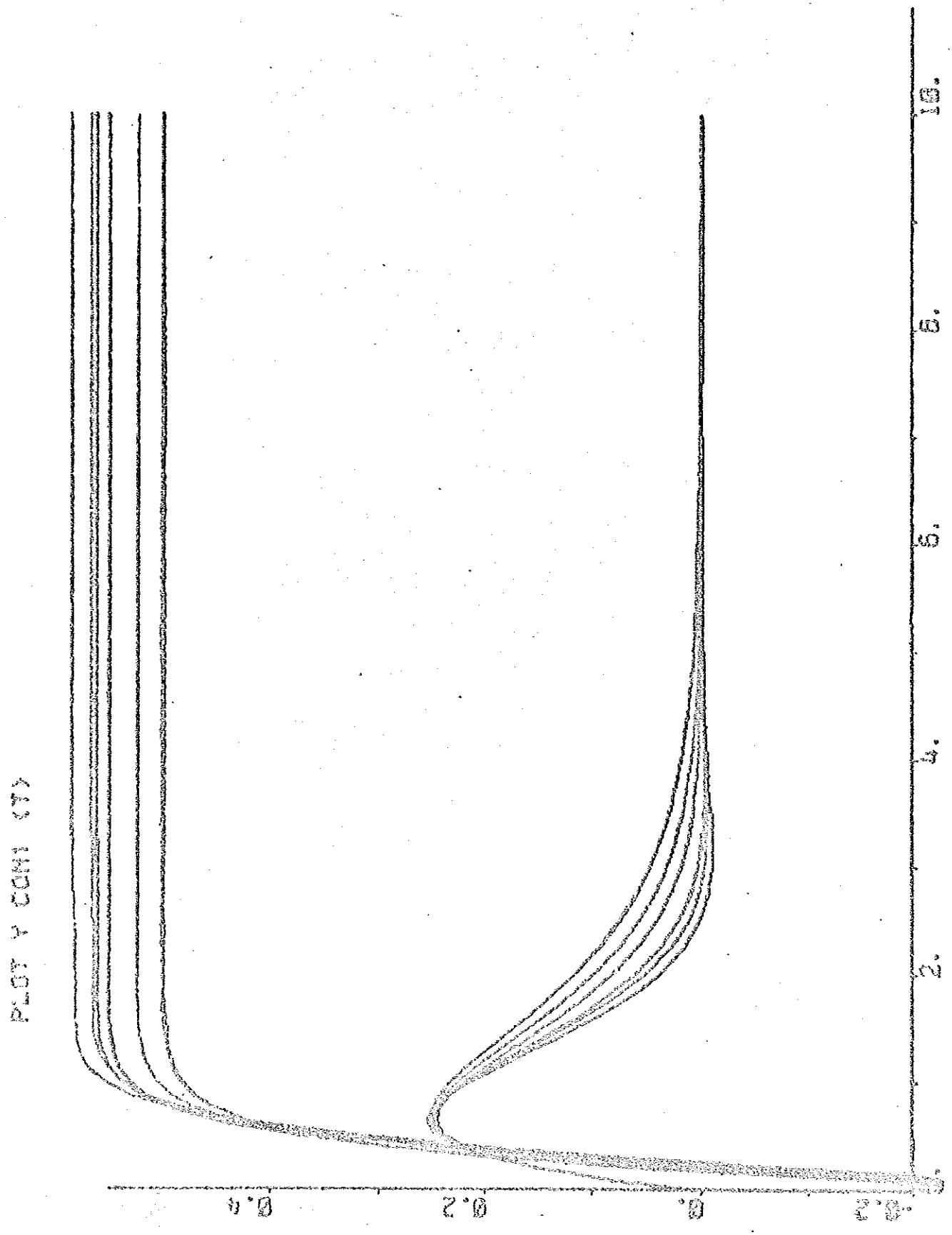


Fig. 8