INTEROPTDYN-SISO

A Tutorial

Polak, Elijah; Åström, Karl Johan; Mayne, David Q.

1982

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

CODEN:LUTFD2/(TFRT-7240)/1-053/(1982)

INTEROPTDYN-SISO: A TUTORIAL

E POLAK
K J ÅSTRÖM
D Q MAYNE

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
MARCH 1982

LUND INSTITUTE OF TECHNOLOGY
DEPARTMENT OF AUTOMATIC CONTROL
Box 725
S 220 07 Lund 7 Sweden

Document name
INTERNAL REPORT

Date of issue
March 1982

Document number
CODEN:LUTFD2/(TFRT-7240)/1-053/(1982)

Author(s)
E Polak
K J Åström
D Q Mayne

Supervisor

Sponsoring organization
STU Contract 77-3548
NSF, SRC

Title and subtitle

INTEROPTDYN-SISO: A TUTORIAL

Abstract

INTEROPTDYN-SISO is an interactive package for design of a special class of single-input single output linear feedback systems. The performance specifications are given in terms of the closed loop step response, frequency response criteria, bounds on plant input and its derivative, and bounds on design parameters. The package is based on INTRAC-C, a University of California, Berkeley, extension of the language INTRAC, from Lund Institute of Technology, the classical design package CDP from Imperial College, and the semi-infinite optimization code OPTDYN, developed at the University of California Berkeley. The package runs under UNIX and produces graphical displays for HP2648A, and TEKTRONIX 4025 black and white terminals and for TEKTRONIX 4027 and RAMTEK color terminals.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
53

Recipient's notes

Security classification

INTEROPTDYN-SISO: A TUTORIAL

by

E. Polak*, K. J. Astrom** and D. Q. Mayne***

ABSTRACT

INTEROPTDYN-SISO is an interactive package for design of a special class of single-input single-output linear feedback systems. The performance specifications are given in terms of the closed loop step response, frequency response criteria, bounds on plant input and its derivative, and bounds on design parameters. The package is based on INTRAC-C, a University of California, Berkeley, extension of the language INTRAC, from Lund Institute of Technology, the classical design package CDP from Imperial College, and the semi-infinite optimization code OPTDYN, developed at the University of California Berkeley. The package runs under UNIX and produces graphical displays for HP2648A, and TEKTRONIX 4025 black and white terminals and for TEKTRONIX 4027 and RAMTEK color terminals.

*Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California, Berkeley, Ca. 94720

**Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

***Department of Electrical Engineering, Imperial College, London, SW7 2BT, England.

CONTENTS

1. INTRODUCTION.

INTEROPTDYN-SISO is an interactive package for the design of single-input single-output (SISO) linear feedback systems. It was developed at the University of California, Berkeley, by extending, modifying and combining INTRAC [A1,W1], an extendible interactive language from, CDP [D1] a SISO classical design package, and a semi-infinite optimization FORTRAN code OPTDYN [B3] which implements the Gonzaga-Polak-Trahan algorithm [G1].The package currently runs on the DEC VAX 11/780 under the UNIX operating system and allows the use of HP2648A and TEKTRONIX 4025 black and white terminals as well as TEKTRONIX 4027 and RAMTEK color terminals. Since almost all of the code in the package is in standard FORTRAN, the package is highly portable.

The package is intended for the design of control systems of the form shown in Fig. 1, i.e., a simple feedback configuration. The controller may be broken up into two parts as shown in Fig. 1.

The control system performance specifications must be given in terms of (i) an envelope on the closed loop step response, (ii) frequency domain criteria, (iii) upper and lower bounds on the amplitude of the plant input and its derivative, resulting from a step input to the control system, and (iv) upper and lower bounds on design parameter amplitudes.

The allowed envelope on the step response shown in Fig. 2.

The frequency domain specifications are in the form of a parabola in the s-plane, to the left of which closed loop poles are to be placed, see Fig. 3, below, and of gain and phase margins. To ensure that the closed loop poles are in the region specified, one plots a modified Nyquist diagram of the return difference evaluated along the parabola in the s-plane and one makes sure that the origin is not encircled in the G(s)-plane by encasing the origin in the G(s)-plane inside a parabolic region defined by the gain and phase margins. To avoid problems caused by "unstable" open loop poles, the return difference is renormalized (see Sec. 2.3). Note that when the parabola in the s-plane degenerates to the jw-axis, the usual Nyquist criterion and gain and phase margins are obtained.

The plant and the compensators and are assumed to be linear. They may be characterized either in terms of the coefficients or in terms of the poles and zeros of their transfer functions. A state space characterization is also available and the command CONVERT may be used for transforming one description into another. However, the design parameters can only be the coefficients of the denominator or the numerator of the compensator transfer functions. The CDP commands for manipulating system descriptions, refer to the plant as SYS, the (continuous) feedforward controller as

CFF, and the (continuous) feedback controller as CFB. We shall use this notation in referring to the appropriate parts of the closed loop control system.

The design problem is reformulated as a semi-infinite optimization problem with inequality constraints, of the form

$$\min\{f(z) \mid g^j(z) \leq 0,\ j = 1,2,\ldots,m;\ phi^k(z,p_k) \leq 0,\ p_k \in P_k,\ k = 1,2,..,l\} \quad (1)$$

The algorithm used in the package for the solution of this problem is the Gonzaga-Polak-Trahan phase I - phase II method of feasible directions for semi-infinite problems (see [G1]). This algorithm has been used successfully in the design not only of control systems, but also of electronic circuits [P2], digital filters [L1], and structures [B2, P1]. To display a simple description of the algorithm, type the command ALGO. To obtain more detailed information on the operations in each step, type the command ALGO STEPn, where n stands for step number.

The upper and lower bounds on the design parameters give rise to the functions $g^i(z)$. The constraints on the open loop frequency response, on the step response, on the plant input and its time derivative give rise to 9 functions $phi^k(z,p_k)$. The built in cost functions f(z) are the integral square error of the closed loop system step response

$$\int_0^{tfinal} [1 - y(z,t)]^2 \, dt \qquad (2)$$

and, for some cases, the integral of the square of the input to the plant u(x,t) due to a step input to the system:

$$\int_0^{tfinal} u(z,t)^2 \, dt \qquad (3)$$

2. HOW TO RUN THE PACKAGE

A short description of how the package may be used is given in this section. The package runs on the VAX 11/780 under the UNIX operating system. The package does not distinguish between upper and lower case letters and the user may issue commands in either form.

2.1 Start Up

After entering the appropriate UNIX directory, the command d.siso starts the package. The package requests data in an interactive mode. To leave the package temporarily, one types CNTRL z. The package is then suspended and one is brought back to UNIX. The UNIX command fg returns the user to the package again.

2.2 Control System Definition

The control system is specified by entering the transfer functions of the plant SYS, and of the compensators

CFF and CFB by means of CDP [D1] procedures which have been converted to INTRAC-C commands. The design parameters in CFF and CFB which are to be adjusted by the optimization program are identified while entering the transfer functions. (The need to enter transfer function coefficients either as numbers or as variable names has necessitated some modification to the procedures in CDP [D1].)

To assist the user in the selection of compensator structure, the package includes the commands NYQUIST, ROOTLOCS and BODE, which request parameters in conversational mode and produce the appropriate plots.

The command ENTER is used to define the transfer functions. Following this command the package will ask for the parameters in a conversational mode. Two dialogues illustrate how the command operates. The numbers entered in these dialogues correspond to the example to be considered in the next section.

Dialogue 1.

This dialogue shows how the command ENTER is used to define the plant. The user's inputs are in capitals:

```
ENTER
Form of data, element
CPY SYS
Gain
1
```

Time delay

0

Order of numerator and denominator

0 3

Numerator coefficients is ascending order

6.0 8.0 5.0 1.0

Denominator coefficients in ascending order

6.0 8.0 5.0 1.0

#

A slightly modified dialogue is used to describe a compensator with adjustable parameters.

Dialogue 2

ENTER

Form of data, element

CPY CFF

Gain

1.0

Order of numerator and denominator

2 2

Numerator coefficients in ascending order

Z(1:1) Z(2:1) Z(3:1)

Denominator coefficients in ascending order

0.0 1.0 0.0

#

The transfer function CFB is entered in a similar way. The feedback compensator is set to unity by default.

The coefficient values that were entered may be checked by the command CHEK; they may be altered by the command MODIFY. The modifications are made conversationally.

2.3. Initialization of the Optimization

Once the system descriptions have been entered the design problem is converted into the form of the optimization problem (1) and the optimization is initialized by the command SISOINIT.

The program defines 2n functions $g^{i}(z)$ from the inequalities: $bl(i) \leq z(i) \leq bu(i)$, $i = 1,2,3,\ldots,n$, where n is the dimension of the design vector, bl(i) is the lower bound on z(i) and bu(i) is the upper bound on z(i). The user may specify one of the following two constraints in the frequency domain: (i) phase and gain margins, which are converted into a forbidden parabolic region in the Nyquist plane from which the ordinary Nyquist locus is to be excluded, or (ii) a parabolic region in the s-plane in which the closed loop poles are to be located, which is translated into a parabolic region in the G(s) plane from which the modified Nyquist locus is to be excluded. Given that G(s) = n(s)/d(s), the ordinary Nyquist locus of G(s) is obtained by plotting [d(s) + n(s)]/d(s) for s = jw, $0 \leq w0 \leq w \leq wc \leq \infty$, while the modified Nyquist locus is obtained by plotting [n(s) + d(s)]/(s + a)**k for s on the parabola in the s-plane, again for $w0 \leq w \leq wc$, with a such -a is inside the parabolic region and k = degree[n(s) + d(s)]. In both cases,

(when the zeros of the denominator of the expression are in the stable region in the s-plane) stability is ensured by requiring that the origin in the G(s) plane not be encircled by the locus. The frequency domain constraints give rise to $phi^1(z,w)$. In addition, the user may specify constraints on the step response: overshoot, rise time and rise amplitude, settling time and settling amplitude, which give rise to the functions $phi^k(z,t)$, with k = 2,3,4,5; the user may also specify upper and lower bounds on the plant input u and its derivative udot (stored in BUU, BLU, BUUDOT, BLUDOT, respectively), which give rise to the functions $phi^k(z,t)$, k = 6,7. Constraints on u and udot can only be specified when the corresponding transfer function relating them to the system input is proper. Finally, the user has a choice of two cost functions: integral square error or integral square plant input, both corresponding to a unit step input to the system, see (2) and (3).

The selection of constraints to be used is achieved by changing an indicator in the symbol table from 0 to 1. The appropriate indicators are NYCON1, NYCON2, STPCON, UCON, UDTCON, for the ordinary Nyquist locus, modified Nyquist locus, step response, plant input, and derivative of plant input (for a step input to the closed loop system), respectively. The names in the symbol table of the constants a and k used for defining the modified Nyquist locus are ACONST and KEXP. The parabola in the s-plane is defined by x =

RLP1*y**2 + RLPO, while the parabola in the G(s)-plane is defined by y = P1*x**2 - PO, with x and y corresponding the the real and imaginary axes, respectively. To select square integral error cost one sets OBJECT = 1, for square integral input cost, one sets OBJECT = 2.

When the command SISO is typed in, the program asks for a description of the specifications in conversational mode, as shown below and sets all constants and variables to appropriate values, so the user need not concern himself with the above described constants in the symbol table at this stage. The user is assisted by (color) diagrams which are displayed on the screen. The following responses are reasonable for the problem defined by the dialogues 1 and 2, above. The user's responses are in upper case.

Dialogue 3

```
   SISOINIT
COMMENT: define bounds on z
   type in bu(1)
   #50.
   type in bl(1)
   #0.
   type in bu(2)
   #50.
   type in bl(2)
   #0.
   type in bu(3)
```

#50.

type in bl(3)

#0.

COMMENT: define an initial parameter vector Z do you wish to use the default values z(i) = 1.?

#NO

type in z(1)

#5.

type in z(2)

#5.

type in z(3)

#5.

COMMENT: define the frequency interval [w0,wc] for the constraints.

do you wish to use the default values w0 = 10E-6, wc = 30 ?

#YES

COMMENT: define the stability constraints.

do you wish to use constraints on the actual Nyquist plot?

#YES

do you wish to use the default gain margin of 2.2 and default phase margin of 45 deg?

YES

do you wish to use constraints on the modified Nyquist plot?

#NO

```
COMMENT: define the time domain constraints.
   do you wish io have constraints on the step response?
   #YES
   please give values for the above diagram
   over = ?
   #1.1
   risamp = ?
   #.7
   setamp = ?
   #.05
   trise = ?
   #.5
   tset = ?
   #2.5
   tfinal = ?
   #5.
   do you wish to have amplitude constraints on the control?
   #NO
   do you wish to have amplitude constraints on udot?
   #NO
COMMENT: select a criterion.
   if you wish to use integral square error criterion, type
   in 1; if you wish to use input energy criterion, type in
   2
   #1
COMMENT: adjust algorithm parameters.
   do you wish to use the default values e = .2, oldstp =
```

100. ?

#YES

#

2.4 The Optimization Algorithm

It is difficult to use this package in an intelligent way without having, at least, an elementary idea of how the optimization algorithm works and how it is implemented in the package. For a complete description of the code and parameters of the Gonzaga-Polak-Trahan algorithm used in this package see [B3]. In brief, the algorithm has the following form:

ALGORITHM:

STEP 0: Initialize the design vector Z, the iteration counter ITER and algorithm parameters E, OLDSTP, ALPHA, BETA, DELTA, PUSH (the optimization problem definition also occurs in this step).

STEP 1: Evaluate the cost and constraint functions.

STEP 2: Determine the E-active constraints and the corresponding gradients.

STEP 3: Evaluate the E-optimality function THETA and the corresponding search direction H.

STEP 4: If THETA < -DELTA*E go to Step 6.

STEP 5: Adjust E or Q, as required.

STEP 6: Compute ZNEW by means of the phase 1 - phase II Armijo stepsize rule.

STEP 7: Set Z = ZNEW.

END.

In Step 2, the algorithm computes the maximum constraint violation

$$PSI(Z) = \max\{0;\ g^j(Z),\ j = 1,2,\ldots,m;$$

$$\max_{p_k \in P_k} phi^k(Z,p_k),\ k = 1,2,\ldots,l\} \qquad (4)$$

It then computes the gradients of those $g^j(Z)$ and $phi^k(Z,p_k)$ which satisfy

$$g^j(Z) \geq PSI(Z) - E,\ j \in \{1,2,\ldots,m\} \qquad (5)$$

$$phi^k(Z,p_k) \geq PSI(Z) - E,\ k \in \{1,2,\ldots,l\} \qquad (6)$$

These gradients are then used to construct the quadratic program which defines THETA, as its value and H (the search direction) as its argmin. The quadratic program has parameters PUSHG (for the g constraints), PUSHF (for the functional phi constraints) and SCALE which can be used to recondition a badly scaled design problem.

The algorithm code is written in FORTRAN. It has a number of break points in it (e.g. COPFE110, QP90) at which it tests a flag and if the flag is set, computation is suspended and the program calls INTRAC-C, as if it were a subroutine. This enables the user to examine the current results of the computation either by printing out values or by plotting graphs, modify parameters in the algorithm or

compensator and carry out whatever diagnostic computation the user may find to be helpful. To find out where the algorithm execution has been suspended, the sophisticated user types the command WHERE. For a list of all break points use the command BREAKS. The breaks can be used to modify the execution of the algorithm. For example, to ensure that the algorithm stops at COPFE110 (before the function evaluations), the user sets a flag by means of the command

```
HALT COPFE110 ALWAYS
```

To continue execution after a stop, the user issues the command

```
GO
```

To re-evaluate all the functions, the user types

```
GO COPFE110
```

For more complex examples of execution control, use the LIST command to display the macros STEP2, STEP3, STEP45, ARMIJO and RUN. These macros largely remove the need for the user to be knowledgeable as to the locations of the break points in the FORTRAN code.

The parameters and variables in the FORTRAN code which the user may need to examine or modify interactively, are stored in the SYMBOL table. They can be listed by means of the command SYMBOL and they can be altered by the command SET. See Section 4 for further details.

The algorithm can cycle in the loop defined by Steps 2 to 5 and in the stepsize calculation in Step 6. We shall refer to this cycling as "inner iterations". The algorithm can be executed in two ways.

(i) One can execute k iterations of the algorithm and store the results, by means of the command

RUN k STORE

which stores Z, PSI and F in the arrays ZG, PSIG and FG (where PSI is as defined above and F is the value of f(Z)). When storage is not desired, but only the values of ITER F PSI THETA and E are to be displayed, replace STORE with PRTALL (ITER is the iteration number).

(ii) One can execute the algorithm almost one Step at a time by means of the commands STEP2, STEP3, STEP45 and ARMIJO (for Step 6), which allow one to inspect its behavior in the inner iterations.

The algorithm normally stops at the end of Step 2 so as to allow one to examine the values of design parameters and corresponding constraints. The macro RUN, ARMIJO, RARMIJO, RARMIJOS and some of the other macros for optimization call the macro SETXBR which detects if any of the algorithm parameters have been changed and returns execution to the correct point in the program. For example, initialization changes Z, E, OLDSTP, and possibly ALPHA, BETA, DELTA and MU, which results in Step 2 being executed one more time

before proceeding. However, SETXBR does not detect if any of the problem parameters have been changed (viz. NYCON1, NYCON2, STPCON, UCON, UDTCON, OBJECT, BL, BU, BLU, BUU, BLUDOT, BUUDOT, PO, P1, RLPO, RLP1, OVER, RISAMP, SETAMP, TRISE, TSET, TFINAL). Consequently, whenever one of these parameters is changed, the user should execute the command STEP2 in order to return to program to the correct point of execution. Otherwise the program may jam.

2.5 Optimization Execution

After the initialization is completed, the optimization is executed by the command

```
RUN ITER STORE
```

The number ITER specifies the number of iterations to be executed; the option STORE causes the values of Z, F and PSI to be stored in the arrays ZG, FG and PSIG, respectively. F is the value of the cost function; PSI is the value of the largest constraint violation. During execution, the results of each iteration are printed as follows:

```
I   F   PSI   THETA   E
```

where I is the iteration number; THETA and E are internal variables related to the optimization algorithm. The value E = 0 indicates that the current design parameters satisfy the F. John optimality condition for semi-infinite programming (see [G1]). All constraints are satisfied if PSI is zero.

In the normal, "time varying" version, the number E is monotonically reduced during the optimization (a "time invariant" version can be created by the command RUN ITER STORE TI). Sometimes the optimization becomes suspended because the quadratic program which computes the search direction fails. When this happens a message appears. To "resuscitate" the program, it is necessary to decrease E in order to reduce the number of active gradients. This can be done by making use of the scratch pad, as follows.

ps EE = E/4

Set E = EE

It is also possible to experience very slow progress in the computation due to a poor initial design or to poor problem scaling. In that case, an experienced designer can execute the algorithm one step at a time using the commands STEPn, n = 2,3,45, and RARMIJOS, and examine the results of the computation at the end of each step. The most interesting information is obtained at the end of Step 3, where one can compute the angles between the computed search direction H and the active gradients by means of the macro PRTANG, and in the step size calculation in Step 6, where the active constraints and the difficulties encountered in step size calculation can be displayed by means of the macro RARMIJOS. When the angles between the search direction H and the active gradients are badly unbalanced due to poor problem scaling, the experienced designer can restore a certain

amount of balance by modifying the PUSHG and PUSHF factors for the optimality function THETA. The push factors are in the symbol table (see [B3]). The macro RARMIJOS displays graphically the inner iterations in the step size calculation. In particular, when this macro is executed, one gets a very good idea as to which of the constraints is causing most of the difficulty as well as to whether the values of ALPHA or BETA are too large. OLDSTP, the first step size tried in the Armijo subprocedure, is automatically adjusted in the course of the computation and usually needs to be set only during the first few iterations. When more than 3 or 4 inner iterations are required for step size calculation, it may be desirable to reduce the parameters ALPHA and BETA. OLDSTP should be increased if the inner iterations were spent in increasing step length and reduced if the inner iterations were used to reduce the initial step length. Caution: The parameters should not be changed frequently or unpredictable algorithm behavior may result. If the jamming in Step 6 is attributable to E being too small, so that a certain constraint is neglected in the calculation of H, E can be increased at this point to improve computational efficiency.

It is not always clear in advance that all the constraints that are specified in the initialization stage can actually be met with the given compensator structure. When the constraints cannot be met, the algorithm will eventually

jam with PSI > 0. At this point, the designer may decide to change the compensator or relax some of the design constraints. As we have mentioned earlier, when a design parameter is changed, it is necessary to execute the command STEP2 before any other command such as RUN or RARMIJOS. The package also includes an interrupt feature, activated by hitting the interrupt key, which can be used to interrupt macros such as RUN.

2.5 Analysis of Results

When the optimization is suspended after the number of iterations specified in the RUN command, the properties of the resulting closed loop system may be investigated. The command

SYMBOL

can be used to find out which of the quantities of interest are in the main symbol table, while the command

PTABLE

can be used to display the quantities stored in the scratch pad symbol table (see Section 4). The command

PRINT

can be used to display any value in either symbol table. First, display PSI: PSI = 0 indicates that all the constraints are satisfied, PSI > 0 indicates that they are not. To find out which constraints are not satisfied, first

display G, to determine if the bounds on the design parameters are violated. Next, PHI is a matrix whose rows are the constraints on the frequency and time domain responses. To find the values of the parameters p at which the maximum occurs in each row, use the command

MATMX V1 V2 = MAX(PHI)

to compute the indices of the maximum row values of PHI in V1 and the corresponding column numbers in V2. Alternatively, use graphics. The command

SISOSTEP col iter k; window

displays the step response for iteration number iter in color col. The variable k should be 1 for the first display. This gives scales. For the following displays k should be set to the display number in order to ensure correct labeling. When window is not specified, the display will be in window WSTEP, otherwise it will be in whatever window is named.

Similarly, the commands

SISOU col iter k; window

SISOUDOT col iter k; window

will display the plant input and its derivatives.

The command

SISOSTAB col iter k

displays the Nyquist or the modified Nyquist curve, whichever one is used in the problem, with col, iter and k as above.

To obtain the final value of the design parameters use the command

PRINT Z

To obtain intermediate values, use the command

PRINT ZG(:iter)

with iter the desired iteration number. To obtain the final value of the compensators that are being designed. use the command CHEK. To obtain the final closed loop transfer function, use the command RESPONSE which will display this transfer function and, in addition, if requested, the response of the final closed loop system to step, ramp, parabolic and sinusoidal inputs.

2.6 Summary.

By using eight commands d.siso, ENTER, SISOINIT, RUN, SISOSTEP, SISOU, SISOUDOT and SISOSTAB, it is possible to carry out simple control system design exercises. A complete example is given in the next section. There are additional commands in the package for more sophisticated displays and diagnostic computations. A list of all available commands is

given in the appendices.

3. AN EXAMPLE

We shall now present a complete example to illustrate how the optimization-based SISO control system design package may be used.

The plant to be controlled has the transfer function

$$G_{SYS}(s) = \frac{1}{(s+3)(s^2+2s+2)}$$

It is desired to find a PID controller which satisfies step response specifications of the form given in Fig. 2, with over = 1.1, risamp = 0.7, setamp = 0.05, trise = 0.5, tset = 1.2 and tfinal = 5. It is also required that the phase margin be 45 degrees and the gain margin be at least 2.2.

The PID regulator has the transfer function

$$G(s) = \frac{Z(1) + Z(2)*s + Z(3)*s^2}{s}$$

where Z(1), Z(2) and Z(3) are the components of the design vector Z; they are constrained to lie between 0 and 50.

To obtain a solution to the design problem the package is initialized as before with the commands d.siso, ENTER, and SISOINIT, as shown in the Dialogues 1-3 in Sections 2.2 and 2.3. The command

RUN 2 STORE

results in the following display on the screen.

I=0.0 F=0.0 PSI=0.0 THETA=0.0 E=0.2

I=2 F=0.248 PSI=0.043 THETA=-0.05 E=0.2

Since PSI is positive, the constraints are not satisfied and the optimization is continued for one more iteration with the command

RUN 1 STORE

The following results are then obtained

I=3 F=0.27 PSI=0.024 THETA=-0.03 E=0.1

The algorithm is unable to solve the quadratic programming problem and a message appears. The current design parameters are

Z = (15.5 19.0 12.9)

The step response (displayed by the SISOSTEP command) has a slight undershoot. (See Figs. 4a and 4b). To continue the optimization the parameter E is changed manually by the command

SET E = 0.02

This reduces the number of active constraints and the optimization can be continued. After 8 iterations we get

I=7 F=0.267 PSI=0.0 THETA=-0.014 E=0.02

I=8 F=0.245 PSI=0.0 THETA=5.7e-5 E=0.02

The design vector is

Z = (22.9 19.0 15.9)

The displays produced by the commands SISOSTEP and SISOSTAB show that all constraints are satisfied.

The standard PID regulator has very high gain at high frequencies. To reduce the high frequency gain of the regulator, the transfer function of the regulator is modified to

$$G_{CFF}(s) = \frac{z1 + z2*s + z3*s^2}{s + 0.1s^2}$$

using the MODIFY command on the file CFF. With the previous value of the design vector the overshoot becomes 25% which is far too high. Running the optimization algorithm for 11 more iterations gives

I=19 F=0.34 PSI=0.05 THETA=-0.07 E=0.0025

Since PSI is not zero the constraints are not satisfied. Analysis of the step response (see Fig. 5b) shows that the overshoot is 15%. The Nyquist curve (see Fig. 5a) also reaches into the forbidden region. The design vector is

Z = (20.8 12.8 15.7)

No substantial improvement is obtained even if the program is run for many more iterations. The conclusion is clear

that the specifications cannot be satisfied with the chosen configuration. Either one must be satisfied with the design obtained or else one may try to change the value 0.1 in the regulator to a smaller value.

4. INTRAC-C

4.1. Introduction

INTRAC [A1, W1] is an extendible, interpreted, BASIC like language. It is a small nucleus, written in FORTRAN, which can be used for converting a set of FORTRAN subroutines into an interactive package. These subroutines are accessed via INTRAC commands. In its nuclear form, INTRAC is a very simple language: it has neither an arithmetic expression parser (so that only binary arithmetic operations are allowed), nor subscripted variables. However, it does have complete logic capability, interrupt capability, an ability to read both numbers and strings inputted from a terminal, as well as to produce both alphanumeric and graphical output, and an ability to control the execution of a FORTRAN program. In addition, it has a macro facility which makes it possible to write programs (macros) (which can call other macros as subroutines up to 7 layers deep) and hence new commands in INTRAC itself rather than in FORTRAN. INTRAC has very nice diagnostic features which make it very easy to write and debug macros. (The run time diagnostic are actuated in this package by the ON command (macro) and

switched off by OFF). INTRAC executes rather slowly, particularly when loops are present (for i = 1 to k) and hence it is not suitable for coding of optimization algorithms or cost and constraint functions. However, it is excellent for writing display macros. (See Appendix B.7 for a summary of INTRAC statements.) On the other hand, commands and routines in FORTRAN are much more difficult to write, debug and implement, but they execute much more rapidly than INTRAC. INTRAC has been used in a number of other packages as well, viz. SIMNON, IDPAC, MVDPAC, POLPAC AND SYNPAC, see [A1].

INTRAC-C is an extension of INTRAC for use with the OPTDYN semi-infinite optimization code. The resulting package is called INTEROPTDYN. To use it for solving a specific problem, FORTRAN subroutines for function and derivative calculation must be written, added to the package and the whole code recompiled.

INTRAC-C includes scratchpad commands for both scalar and matrix calculations, and elementary commands for color graphics (see [B1,B3]). The scratch pad commands can be used in macros which can call other macros as subroutines, but it does not allow the use of internal variables.

INTEROPTDYN makes use of three symbol tables: one for the FORTRAN variables and parameters which appear in OPTDYN and the user supplied FORTRAN subroutines, one for the scratch pad variables and one for the INTRAC variables. The contents of the FORTRAN symbol table are displayed by the

command SYMBOL, those of the scratch pad by the command PTAB and those of INTRAC by the command WRITE. All INTRAC-C variable names must end in ., e.g., x., y.. For a FORTRAN or scratch pad variable to be used in an INTRAC-C expression, it must first be transferred to the INTRAC-C symbol table by means of the TRANS command. When a FORTRAN variable, say ALPHA, is transfered to the INTRAC-C symbol table by means of the TRANS command, it becomes ALPHA. in the INTRAC-C symbol table. The same holds true for variables which are transfered into the INTRAC symbol table from the scratch pad symbol table. No special command is necessary for using FORTRAN or INTRAC-C variables in scratch pad expressions. Arithmetic statements in INTRAC-C are preceded by LET, while those in the scratch pad by PSCAL; matrix expressions in the scratch pad must be preceded by PMAT. Scratch pad and INTRAC-C expressions (other than SET) cannot be used to assign values to FORTRAN variables and scratch pad expressions cannot be used to assign values to INTRAC-C variables. When an attempt to make an illegal assignment is made, an error message appears. Thus, the FORTRAN and INTRAC variables are protected from accidental alteration in the scratch pad. The arithmetic capability of INTRAC-C is normally not used in INTEROPTDYN so as to avoid confusion. It is mostly useful when it is essential to create internal or logic variables, which is not possible in the scratch pad. INTEROPTDYN-SISO is an extension of INTEROPTDYN for SISO control system design which was obtained by (i) augmenting

the FORTRAN symbol table to include the parameters needed for control system design specification and (ii) by adding to INTRAC-C all the commands for SISO control system specification, manipulation, computation and display that are contained in CDP [D1] package. The cost and constraint functions for control system design are evaluated by FORTRAN subroutines, some of which were obtained from CDP.

For a complete listing of all the available INTRAC-C commands, see Appendices A and B; [W1] is an INTRAC language manual which is summarized in Appendix B.7.

INTEROPTDYN-SISO includes an extensive library of macros which combine the various elementary INTRAC-C commands into higher level commands. Some of the commands implemented as macros are discussed below.

4.2. Macros for optimization

The flow of the optimization execution is controlled by macros. These can be used to execute one step of the algorithm at a time, to run a given number of iterations and store the results, to perform diagnostic calculations, to display graphically the behavior of the algorithm as it cycles in inner iterations, and so forth. A full list of these macros is given in Appendix C.

4.3. Macros for graphics

The package contains macros for general purpose graphics, e.g., for window selection and for array row or column

plotting, with zoom capability, and labeling, as well as for displays that are specific to SISO design, e.g. of Nyquist plots and step responses. Root locii, Nyquist plots and Bode plots can also be plotted via elementary commands based on CDP [D1] graphics. A full list of macros for graphics is given in Appendix D.

4.4. Macros for matrix calculations.

The scratch pad commands of INTRAC-C allow both scalar and matrix unitary (e.g. ps x = pwr(y 3)) and binary expressions (e.g. pmat A = B + C). These expressions can be used in macros, both to create a still higher level language which eliminates the need to declaring matrix dimensions and for creating arrays and other data for graphical displays. Unfortunately, scratch pad commands do not allow the use of internal variables in macros, which can become a problem when several layers of calls to other macros are used in a macro. To get around this shortcoming we recommend the use of a special convention: names such as ox, oy oox, ooy, for "internal" variables and x, y for "global" variables. As has already been pointed out, the scratch pad has its own symbol table. It can read all the quantities in the main symbol table, which contains the results of the computation of the optimization program and its parameters, but it cannot alter the entries in the main symbol table. Thus, the main symbol table is protected from inadvertent alteration in the process of diagnostic calculations. The SET command must be

used to transfer values from the scratch pad symbol table to the main symbol table. A full list of available macros for matrix calculations is given in Appendix E. Note also that the macros for graphics also make heavy use of scratch pad commands.

ACKNOWLEDGEMENT:

This research was supported by the National Science Foundation under grants ECS-79-13148 and CEE-81-05790, the Joint Services Electronics Program under grant F49620-79-C-0178, The Swedish Board for Technical Development under contract 77-3548, and the U.K. Science Research Council. The programming at Berkeley was done by M.A. Bhatti, T. Essebo, E. Eschen, W. Nye, E. Polak, and A.L. Tits.

REFERENCES

[B1] Bhatti, M. A., Essebo, T., Nye, W., Pister, K. S., Polak, E., Sangiovanni-Vincentelli, A., and Tits, A., "A Software System for Optimization Based Interactive Computer Aided Design", Report No. UCB-ERL M80/14, Electronics Research Laboratory, University of California, Berkeley, 1980. Proc. IEEE I.S.C.A.S. Houston Tx., April 1980.

[B2] Bhatti, A., Pister, K. S., and Polak, E., "Optimal design of an earthquake isolation system", Proc. IUTAM Symp. on Structural Control, Univ. of Waterloo, Waterloo, Ont., Canada, June, 1979.

[B3] Bhatti, M. A., Polak, E., and Pister, K. S., "Optdyn - A General Purpose Optimization Program for Problems With or Without Dynamic Constraints", Report No. UCB/EERC - 79/16, Earthquake Engineering Research Center, University of California, Berkeley, July 1979.

[D1] Daly, K. C., and Katzberg, P., "COSDIC documentation: The Classical Design Suite", Publication No. 73/18, Department of Computing and Control, Imperial College, London SW7 2BT, U.K., June 1973.

[G1] Gonzaga, C., Polak, E., and Trahan, R., "An Improved Algorithm for Optimization Problems with Functional Inequality Constraints", IEEE Trans., Vol. AC-25, No. 1, 1980.

[L1] Lee, T. P., Nye, W. T., and Tits, A. L., "The Design of Digital Filters Using Dynamic Optimization", Proc. 20th IEEE CDC, Dec. 16-18, San Diego, Ca.

[P1] Polak, E., Pister, K. S., and Ray, D., "Optimal Design of Framed Structures Subjected to Earthquakes", Eng. Optimization, Vol. 12, 1976.

[P2] Polak, E., "Algorithms for a class of computer aided design problems: a review", Automatica, Vol. 15, pp 531-538, 1979.

[W1] Wieslander, J., and Elmquist, H., "INTRAC a communication module for interactive programs: Language Manual", CODEN LUTFD2/(TFRT-3149)/1-060/(1978), Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, August 1978.

APPENDIX A: COMMANDS FOR CONTROL SYSTEM

The commands needed for SISO design via interactive optimization fall into two categories: those for entering and examining the design problem and those used in optimization.

For further information on the commands ENTER, CONVERT, CHEK, MODIFY, ROOTLOCS, NYQUIST, RESPONSE and BODE type in the command and then ???

A.1. INTRAC-C commands for SISO Control System Manipulation.

SISOBD: displays the block diagram of the control system to be designed. Be sure to type in GRINIT to initialize the graphics before using this command.

ENTER: to be used for entering system and compensator coefficients. Note: the coefficients to be optimized must be entered as z(1:1), z(2:1) z(3:1),....

CONVERT: to be used for converting system or compensator from one form to another.

CHEK: to check the data describing system part.

MODIFY: to modify part of system or compensator description.

ROOTLOCS: to plot root locus and display pole placement region.

NYQUIST: to plot Nyquist diagram.

BODE: to plot Bode diagrams.

RESPONSE: to plot step, ramp, parabola and sin responses.

A.2. Macros for Optimization.

SISOINIT: initializes design parameters and defines constraints.

RUN k STORE: will execute k iterations of the optimization algorithm and store the results.

SISOSTEP c i k: displays step response in color c corresponding to design parameter values at iteration i, with k indicating the number of the graph plotted.

SISOU c i k: displays plant input in color c corresponding to design parameter values at iteration i, with k indicating the

number of the graph plotted.

SISOUDOT c i k: displays time derivative of plant input in color c corresponding to design parameter values at iteration i, with k indicating the number of the graph plotted.

SISOSTAB c i k: displays nyquist or modified Nyquist plot in color c corresponding to design parameter values at iteration i, with k indicating the number of the graph plotted.

SISOLBLN c i k;j: labels nyquist plot in color c corresponding to frequency point i. When i and j are given, will label j points beginning with i th (the frequence range w0 to wc is divided into q points).

STPCNSTR c : will draw the step constraints in step response diagram in color c. Black will erase.

MARGINS: will enable you to reset both phase and gain margins by recomputing and reseting P0 and P1.

PARABOLA c MOD; y: will draw constraint parabola in Nyquist plain in color c. If y is typed in, it will recompute the parabola from the new gain and phase margins. (The option MOD is needed to make this macro usable in another macro, as well).

APPENDIX B: INTRAC-C COMMANDS

B.1. Commands for control flow

ALGO - Displays program structure and associated breakpoints

BREAKS - Displays a list of all breakpoints

WHERE - Displays name of breakpoint

HALT - Sets up halt condition at specified breakpoint

GO - Transfers control to optimization program

B.2. Commands for diagnostics

SWITCH ECHO ON/OFF - enables/disables echoing of commands

SWITCH TRACE ON/ OFF - enbales/disables echoes of execution of commands

VAXDEBUG FILE ON/OFF - enables/disables file handling trace (off is default condition).

VAXDEBUG FILEDUMP ON - will produce one snapshot dump on fort.7 of filehandler internal data (to be used if file handling seems in error).

B.3. Commands for manipulating variables in the symbol table

PRINT - Displays a variable from the symbol table or the scratchpad

SET - Changes the value of a single variable in the symbol table

SETDIM - Changes actual dimensions of a variable in the symbol table

TRANS - Transfers value of symbol table variable to INTRAC

CHECK - Checks if a variable has been changed by SET

CLEAR - Clears flag used in CHECK

SYMBOL - Displays symbol table

B.4. Commands for the Scratchpad

GETDIM - Returns actual array dimension from symbol table

PDIM - Creates a variable in external symbol table (scratch pad)

PREM - Removes a symbol from the scratch pad

PTAB - Displays external symbol table (scratch pad)

PSCAL - Scalar operations in the scratchpad

PMAT - Array operations in the scratchpad

B.5. Miscellaneous Commands

COPY - Copies a macro file

DELETE - Deletes a macro file

ED - Brings in an editor containing most of the UNIX ex commands for editing macro files.

HELP - Explains usage of the commands

LIST - Lists a macro file on the terminal or deposits it in the file fort.11.

CSH - This command makes it possible to call the shell and execute any UNIX command from the package

B.6. Commands for Graphics

COLOR - Sets color for subsequent graphics output.

CURSOR - Moves cursor to x,y coordinate in preparation for text output

CURSOREL - Positions cursor a specified number of character size units away from x,y coordinate.

CURVE - Draws curve specified by an array.

DEFINE - Defines rectangular windows on screen by a user specified name.

DRAW - Draws vector from previous 'MOVE'ed position to x,y coordinate

ERASE - Erases the whole screen or just a specified window

GRINIT - Graphics initialization. Must be given before doing any graphics. The first time this command

is given, the terminal type is requested.

MOVE - Moves cursor to x,y coordinate in preparation for a DRAW.

PARCURVE - Draws curve in parametric form.

TEXT - outputs strings or numeric values at the position of the graphics cursor. A CURSOR or CURSOREL command must precede a TEXT command.

VECTOR - Draws a vector between specified starting and ending coordinates.

WINDOW - Enters specified window so that 0.0 to 1.0 coordinates appear only in the previously defined rectangular window.

B.7. Summary of INTRAC Statements

These statements are used in writing macros. For details, see the INTRAC language manual [W1].

MACRO <macro identifier>[<formal argument>|<delimiter>|<termination marker>]*
Begins a macro definition and creates a macro. The delimiter is a symbol such as (,*,+,#, etc. The termination marker is a semicolon and is used to separate groups of optional arguments.

FORMAL {<formal argument>|delimiter|<termination marker>}*
Declares formal arguments in a macro definition and when creating a macro.(termination marker = ;).

END
ends a macro and ends macro creation mode. Deactivates suspended macros.

LET {<variable>=}*{<number>|<pad variable>[{+|-|*|/}<number>|<pad variable>] number>|<pad variable>]|<identifier>[+<integer>]|<delimiter> |<unassigned variable>
Assigns (allocates) variables.

DEFAULT {<variable>=}* <argument>
Assigns a variable if it is unassigned or does not exist previously.

LABEL <label identifier>
Defines a label.

GOTO <label identifier>
Makes unconditional jump.

If <argument> {EQ|NE|GE|LE|GT|LT} <argument> GOTO <label identifier>
Makes a conditional jump. (argument is an INTAC variable).

FOR <variable> = <number> to <number> [STEP <number>]
Starts a loop.

NEXT <variable>
Ends a loop.

WRITE [(LP)] [<variable>|<string>]
Writes variables and text strings in ' ' or displays currently available variables. (LP) (or (lp)) option causes the string to be written in the file fort.8 for later print-out.

READ { {<variable> {INT|REAL|NUM|NAME|DELIM|YESNO} }| <termination marker>}*
Reads values for variables from the terminal. (termination marker = ;).

SUSPEND
Suspends the execution of a macro.

RESUME
Resumes the execution of a macro.

SWITCH {EXEC|ECHO|LOG|TRACE} {ON|OFF}
Modifies switches in INTRAC.

FREE { {<global variable>}* | *.*}
Deallocates global variables.

STOP
Causes exit from package.

B.8. Macros for Obtaining Help

HLPEX - Displays the list of macros for execution.

HLPGR - Displays the list of macros for graphics.

HLPPAD - Displays the list of macros for matrix calculations,

APPENDIX C: MACROS FOR OPTIMIZATION EXECUTION

The following is a list of macros which are used either directly or as subroutines for the control of the optimization flow and information monitoring. They are classified as either being primary or subroutine in nature.

C.1. Primary Macros

ARMIJO	executes one iteration with Armijo stepsize calculations display
OFF	turns off diagnostic display
ON	turns on diagnostic display
PRTALL	prints f psi e theta
PRTANG	prints angles between search direction and active gradients
PRTFPSI	prints f and psi
RARMIJO N	executes N iterations with Armijo stepsize calculations display
RARMIJOS N	executes N iterations with Armijo stepsize calculations display and stores results by means of STORE
RUN N; OPTION1; OPTION2	..where N = no. of iterations to perform, OPTION1 can be STORE, PRTALL, PRTFPSI, and OPTION2 can be TI
STEP2	executes Step 2 of algorithm
STEP3	executes Step 3 of algorithm
STEP45	executes Steps 4 and 5 of algorithm
STORE	stores f psi and z in the arrays fg, psig and zg
TI	turns the algorithm into a time invariant version
TYPE	prints information normally displayed by ARMIJO graphically

C.2. Subroutine Macros

ANG
BARMIJO

```
CHECK
CLR
EPS
EXEC
GR
MESSAGE
NIL
SCAL
SETXBR
SKIP
STANDARD
STARM
START1
```

APPENDIX D: MACROS FOR GRAPHICS

In the macros listed below, the following notation and mnemonics are used:

V denotes a column vector.

H(I:) is the ith row of H.

H(:I) is the ith column of H.

ROW denotes operation on a row of a matrix.

COL denotes operation on a column of a matrix.

C, C1, C2 are colors.

I J denote the first and last index of array to be plotted NS denotes no internal scaling: precomputed scaling information must be in the form oVMAX oVMIN.

YESNO refers to asterisk on graph: y YES, n NO.

AXES C.....Plots axes in color C.

BARCOL H(I:) C; I J; TOP BOT.. Computes scale when TOP BOT are not given and barcharts

BARMIJO..... Produces bar charts for Armijo step size calculations when both conventional and functional constraints are to be plotted.

BARG..... Produces bar charts for Armijo. step size calculation when only conventional constraints are to be plotted.

BARROW H(I:) C; I J; TOP BOT.. Computes scale when TOP BOT are not given and barcharts.

BARS V C; I J; TOP BOT.. Computes scale when TOP BOT are not given and barcharts.

BOX..... Draws box in prespecified color.

GRAPHO I..... Produces Armijo stepsize information when STORE is not used.

GRAPHOS I..... Same as above to be used with STORE.

GRAPHPSI YESNO; RUN..... Plots values in PSIG; PSIT.

LBL V C; K; I J..... Computes max and min of els in V in the range I J and writes the element values K units to left of y-axis. (Negative K is ok)

LBLCOL H(:I) C; K; I J.. Same as above.

LBLROW H(I:) C; K; I J...Same as above.

LBLMNMX C; K;... ..... Labels a graph with min and max values produced by PRESCALE.

LBLX C N1 N2 D;Y;X,YESNO... Labels x-axis from N1 to N2, in increments D, Y X are shift parameters, n omits first label

LINE CLR LVL; SLP..... Draws line in color CLR through level LVL default SLP = 0. (slope).

LINA ANGLE C..... Draws horizontal line at angle value.

LVL V C L; I J; YESNO; INCR.. Computes level L for vector V and plots it in color C; if yes, will label line as x-axis with default incr(ement) = 5.

LVLCOL H(:K) C L; I J; YESNO; INCR... same as above

LVLROW H(K:) C L; I J; YESNO; INCR... same as above

PLTBCOL H(:I) C1 C2; I J; TOP BOT... Same as PLT followed by BARS

PLTBROW H(I:) C1 C2; I J; TOP BOT... Same as above.

PLTBMXR H C1 C2; I J... Plots and barcharts maximum, by rows, of matrix H. Scales may be produced by PRESCALE.

PLTCOL H(:I) C YESNO; I J; TOP BOT... see PLT

PLTMXC H C YESNO; I J; TOP BOT...Same as PLOt for maximum by columns of matrix H, scales may be precomputed by PRSECALE.

PLTMXR H C YESNO; I J; TOP BOT... same as PLOT for maximum by rows of matrix H, scales may be precomputed by PRESCALE.

PLTROW H(I:) C YESNO; I J; TOP BOT ... see PLT

PRESCALE H; I J..... Computes the min and max elements of a matrix H between the I-th and J-th columns.

UNIVECT ; N..... .. Computes a vector ONE of length N with 1 as all the elements.

UP;K..... Opens up K lines for text.

WB..... Window: bottom third of screen.

WBE..... Erase WBE.

WC..... Window: center of screen.

WCE..... Erase WCE.

WL..... Window: lower half of screen.

WLE..... Erase WL.

WM..... Window: middle third of screen.

WME

WQ1..... Window: bottom quarter of screen.

WQ1E
WQ2
WQ2E
WQ3
WQ3E
WQ4
WQ4E
WT..... Window: top third of screen.
WTE
WTOPCR..... Window: small upper right corner.

WU..... Window: upper half of screen.

WUE

APPENDIX E: MACROS FOR MATRIX CALCULATIONS

This is a list of macros used for matrix manipulation. names ending in g are single precision for use in graphics.

ANG C = ANGLE(U V)	COMPUTES ANGLE IN DEGREES, COL. VECTORS
COL V = B(:J)	SET V EQUAL TO JTH COL OF B
COLCLIP V = B(:I J)	CLIP OUT COLUMNS I TO J
CON C = CON(A)	COMPUTES CONDITION NUMBER
DET C = DET(A)	COMPUTES DETERMINANT
EIG V L = EIGEN(A)	COMPUTES EIGENVECTORS V EIGENVALUES L
EQ C = B	SET C EQUAL TO B
INV C = INV(A)	COMPUTES INVERSE
MAT C = A op B	WHERE A, B ARE MATRICES AND op is + - * ^ (^ stands for multiplication of B by scalar A)
MATMNG V1 V2 = MAX(Q);I J...Q	IS A MATRIX, TO BE TRUNCATED TO COLUMNS I TO J, V1 IS A VECTOR OF MIN ELEMENTS TAKEN OVER ROWS, V2 IS A VECTOR OF CORRESPONDING COLUMN INDICATORS
MATMX V1 V2 = MAX(Q);I J...Q	IS A MATRIX, TO BE TRUNCATED TO COLUMNS I TO J, V1 IS A VECTOR OF MAX ELEMENTS TAKEN OVER ROWS, V2 IS A VECTOR OF CORRESPONDING COLUMN INDICATORS
MN M I = MIN(V)	V IS A VECTOR, M ITS MIN ELEMENT, I THE CORRESPONDING INDEX
MNG M I = MIN(V)	V IS A VECTOR, M ITS MIN ELEMENT, I THE THE CORRESPONDING INDEX
MCOL C(:I) = B(:J)	SET ITH COL OF C EQUAL TO JTH COL OF B
MROW C(I:) = B(J:)	SET ITH ROW OF C EQUAL TO JTH ROW OF B
NRM C = NORM(V)	COMPUTES NORM OF V

MX M I = MAX(V)	V IS A VECTOR, M ITS MAX ELEMENT, I THE CORRESPONDING INDEX
ROW V = B(J:)	SET V EQUAL TO JTH ROW OF B
ROWCLIP V = B(I J:)	CLIP OUT ROWS I TO J
SP C = <A B>	SCALAR PRODUCT
TRC C = TRACE(A)	COMPUTES TRACE
TRS C = TRANS(A)	COMPUTES TRANSPOSE
DR C = RAD(A)	CONVERTS DEGREES TO RADIANS
RD C = DEG(A)	CONVERTS RADIANS TO DEGREES

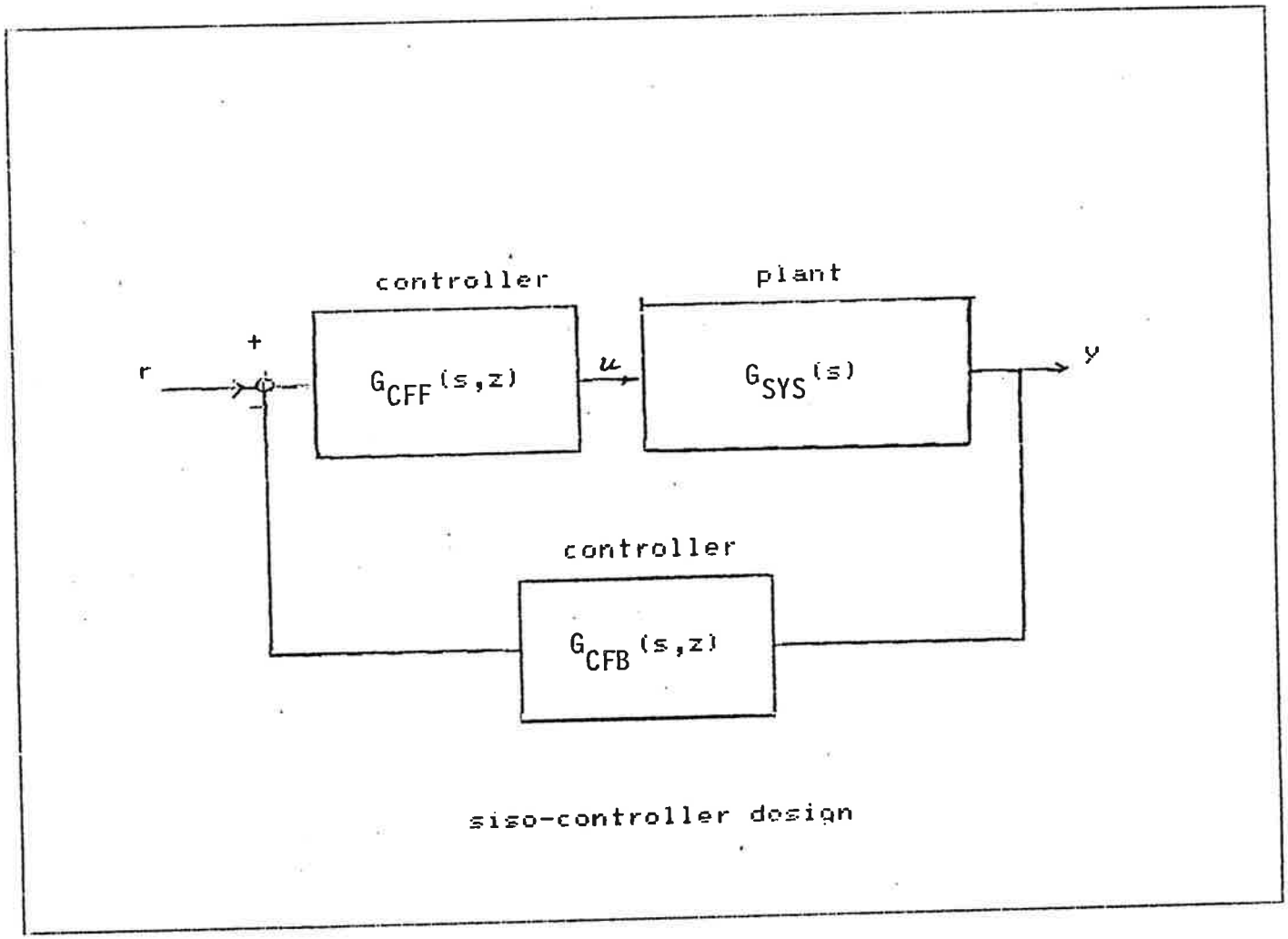


Figure 1. - The system structure allowed by the package.

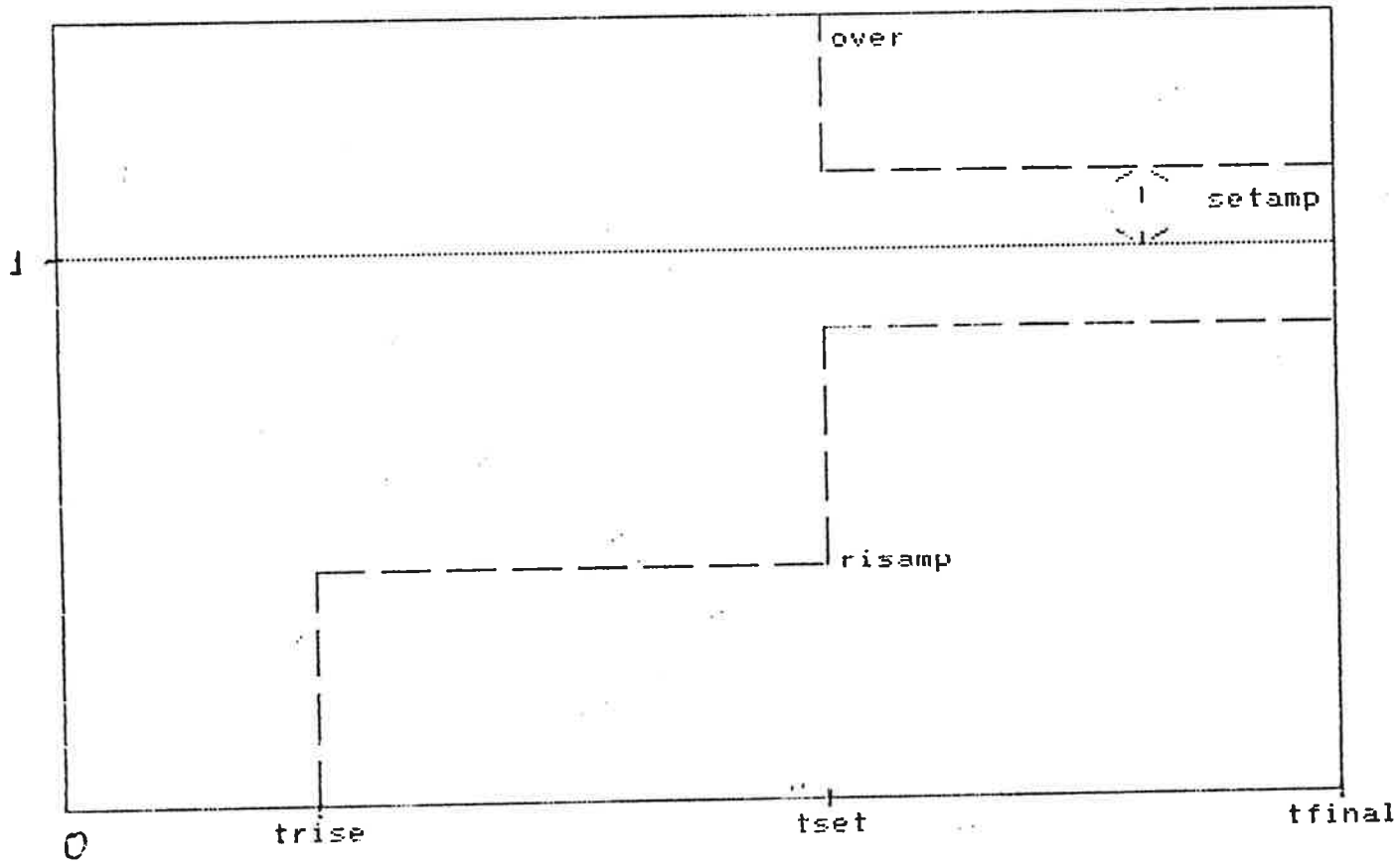


Figure 2. - Time domain specifications.

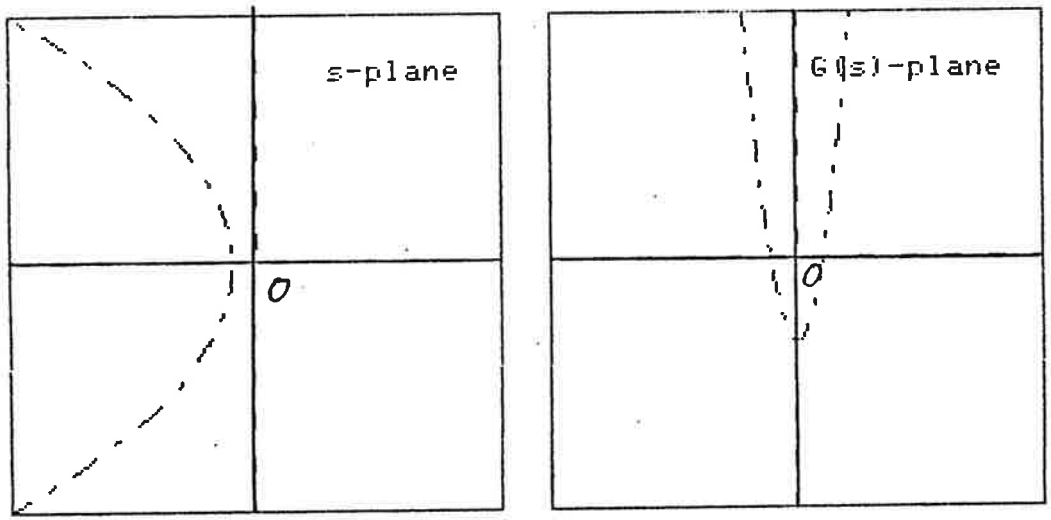
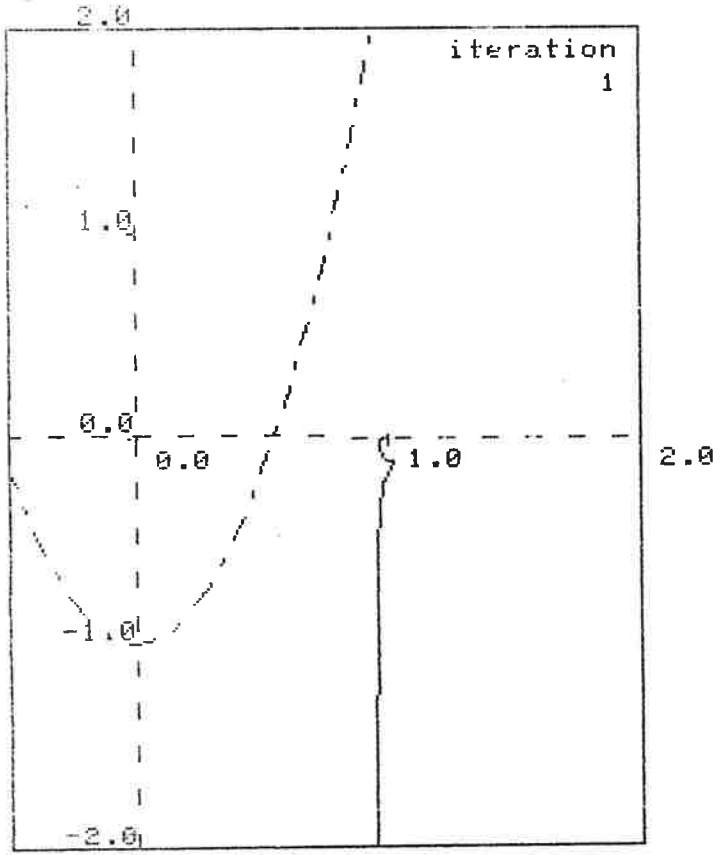
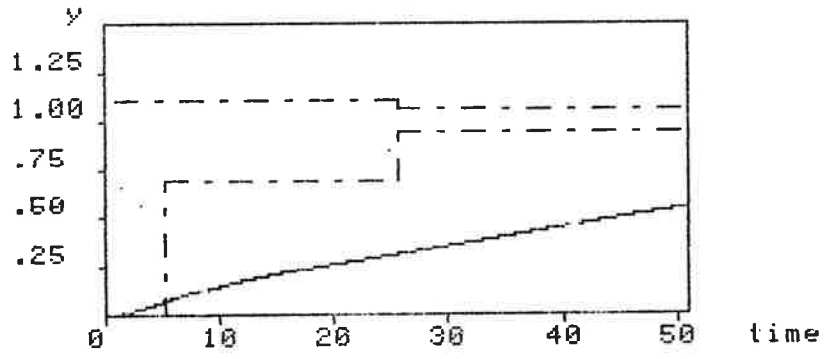


Figure 3. - Frequency domain specifications.

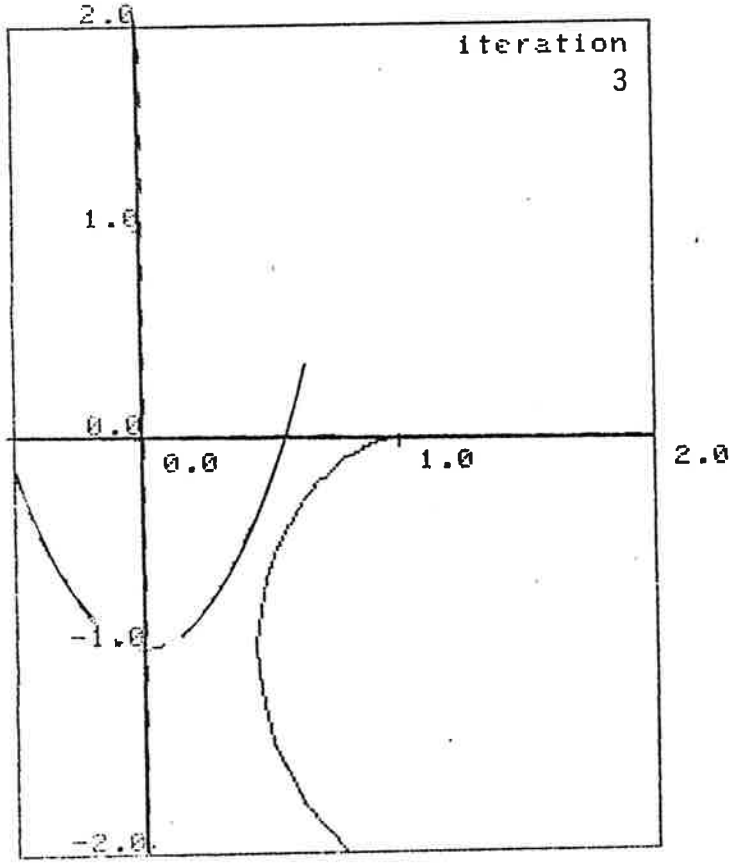


(a) Modified Nyquist Plot

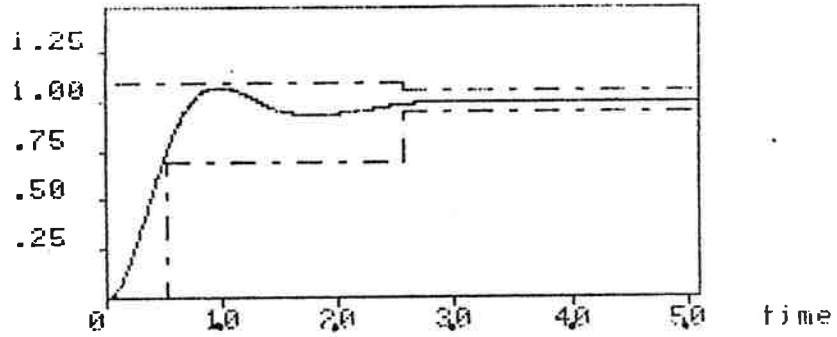


(b) Step response

Figure 3'. - Results after 1 iterations.

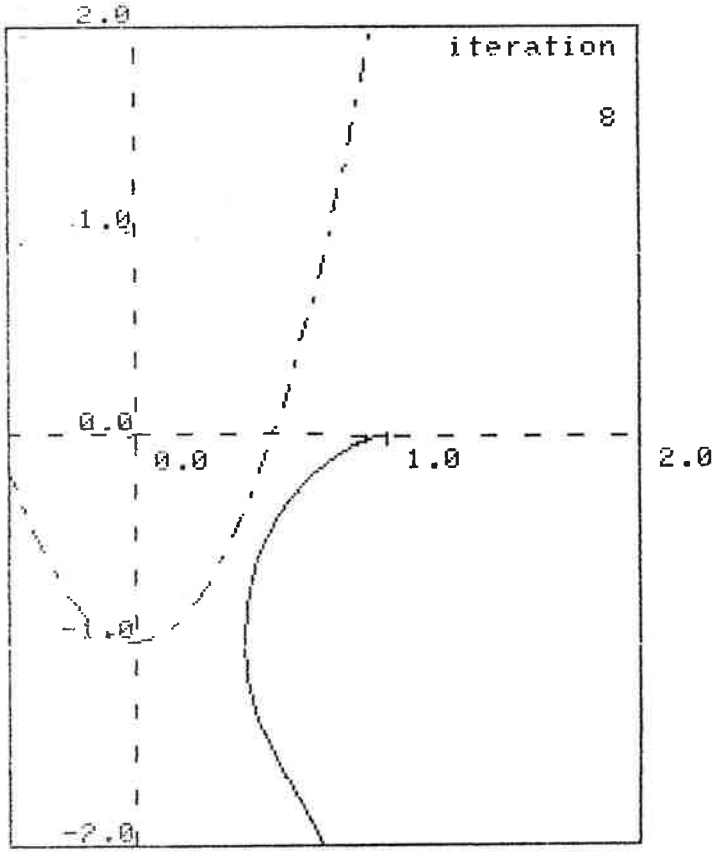


(a) Modified Nyquist Plot

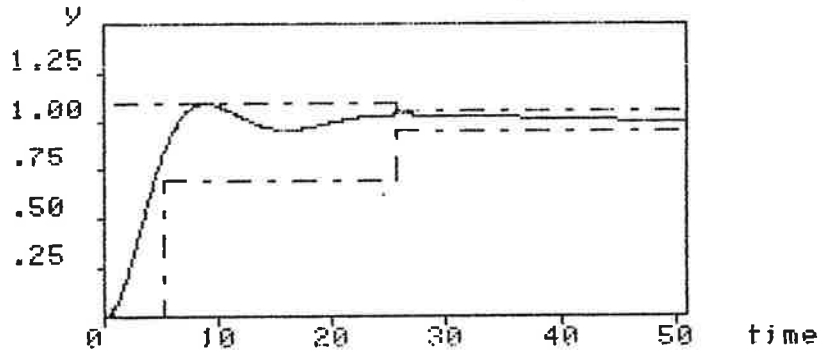


(b) Step response

Figure 4. - Results after 3 iterations.

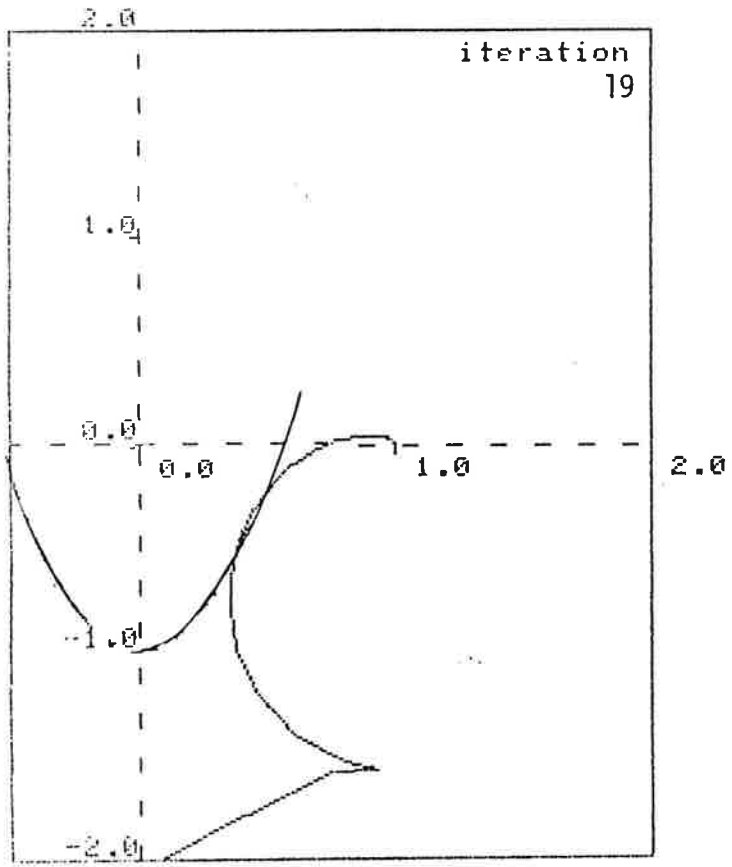


(a) Modified Nyquist Plot

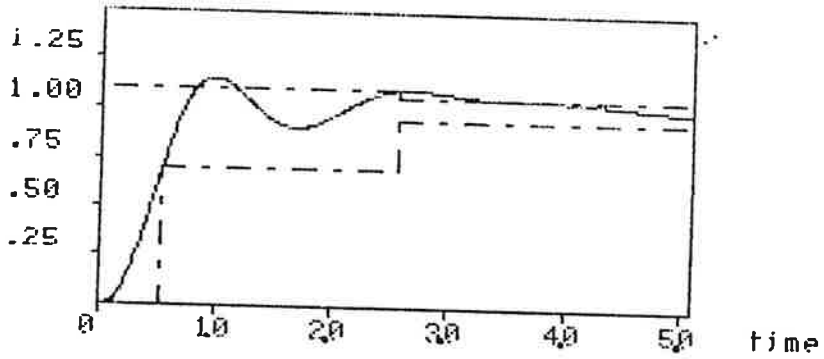


(b) Step response

Figure 4'. - Results after 8 iterations.



(a) Modified Nyquist Plot



(b) Step response

Figure 5. - Results after 19 iterations.