Copyright © 1981, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

INTEROPTDYN-SISO: A TUTORIAL

by

E. Polak and K.J. Astrom

Memorandum No. UCB/ERL M81/97

11 December 1981

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

INTEROPTDYN-SISO: A TUTORIAL

by

E. Polak[†] and K. J. Astrom[††]

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 parameter amplitudes. 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.

CONTENTS

[†]Department of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California, Berkeley, California 94720.

[††]Department of Control, Lund Institute of Technology, Lund, Sweden.

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, and extendable interactive language from the Lund Institute of Technology, CDP a SISO classical design package from Imperial College and a semi-infinite optimization FORTRAN code OPTDYN [B3] developed developed in Berkeley. The Berkeley extension of INTRAC is called INTRAC-C; the optimization code implements the Gonzaga-Polak-Trahan algorithm [G1].The package currently runs on the CDC 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, ie. a simple feedback configuration.

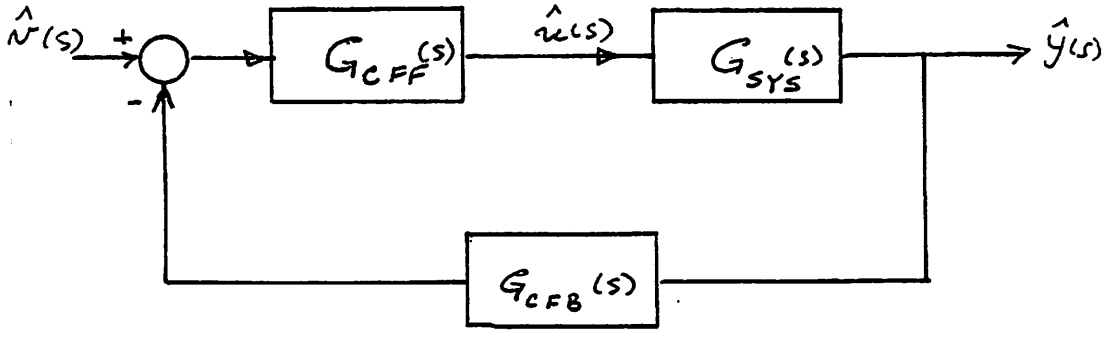

Figure 1. - The system structure allowed by the package.

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

The controller is split up into two blocks CFF and CFB which are called the feedforward and the feedback compensators. Note that this nomenclature does not agree with the standard terminology. Also note that Fig. 1 is not the most general structure for a single-input single-output feedback system.

Specifications may be given both in the time and the frequency domains. The time domain specifications must be expressed in terms of the envelope on the step response shown in Fig. 2.

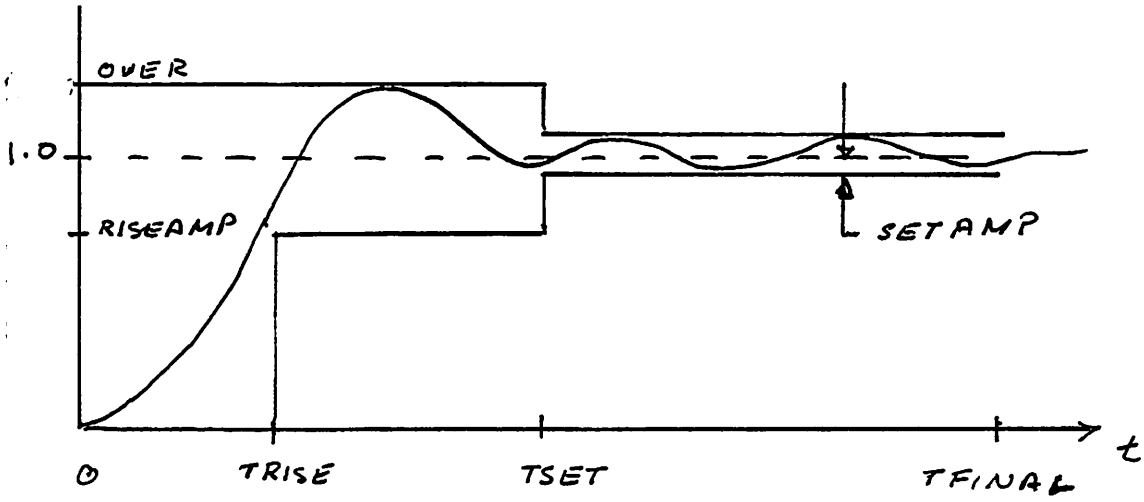

Figure 2. - Time domain specifications.

The frequency response specifications must be given as a gain margin and a phase margin. These values are used by the program to calculate a parabola in the G(jw)-plane as is indicated in Fig. 3. The optimization algorithm will attempt to keep the Nyquist curve outside of the region enclosed by this parabola and will therefore attempt to ensure that the specified gain and phase margins are obtained.

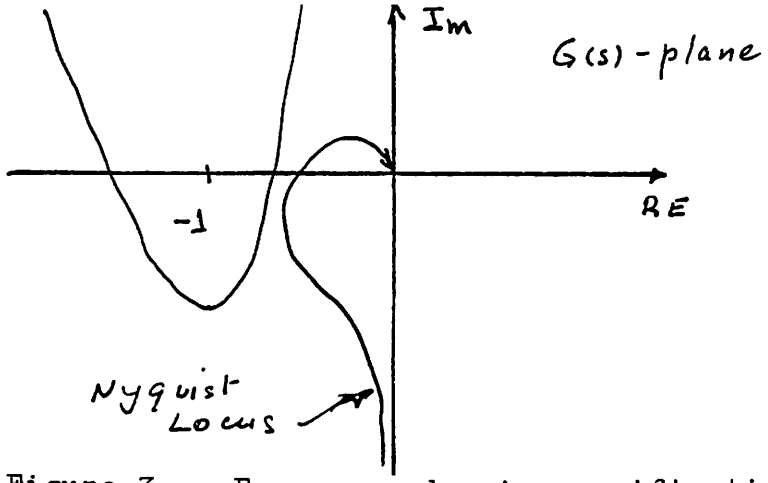

Figure 3. - Frequency domain specifications.

The plant SYS and the compensators CFF and CFB are assumed to be linear. They may be characterized either in terms of the coefficients of their transfer functions 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 design problem is formulated as a semi-infinte optimization problem with inequality constraints, of the form

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

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 constraints on the open loop frequency response and on the step response are expressed in terms five functions $phi^{k}(z,p_{k})$. The upper and lower bounds on the design parameters are expressed in terms of the functions $g^{j}(z)$. The built in cost function f(z) is the integral square error of the closed loop system step response.

In a modified version of the package, it is possible to introduce constraints on the locations of the closed loop poles: one may require that the poles be to the left of a given parabola in the s-plane. For some controller configurations, The modified program allows minimization of the integral mean square of the input, as well as of the step response. In addition, constraints on the amplitude of the plant input, on the amplitude of the derivative of the plant input, and a lower bound of the return difference may be specified in some cases.

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 compensators CFF and CFB by means of CDP procedures which have been converted to INTRAC-C commands. The design parameters 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.

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 values which are introduced may be checked by the command CHEK. The values may be altered by the command MODIFY. The changes can be made conversationally.

2.3. Initialization of the Optimization

Once the system descriptions have been entered the optimization is initialized by the command SISOINIT (SISOIN

with over=1.1, risamp=0.7, setamp=0.05, trise=0.5, tset=1.2 and tf=5. It is also required that the phase margin be 45 deg 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 the solution the package is initialized as before with the commands d.siso, enter SYS and CFF, and sisoinit. The command

RUN 2 store

results in the following printout 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

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. The response of the final system to step, ramp, parabolic and sinusoidal inputs can be obtained by making use of the command RESPONSE.

2.6 Summary.

By using six commands d.siso, enter, sisoinit, run, sisostep and sisonyq 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

A complete example is given to illustrate how the optimization 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 the step response specifications of the form given in Fig. 2,

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 n
```

displays the step response for iteration number iter in color col. The variable n should be 1 for the first display. This gives scales. For the following iterations n should be 2.

The command

```
SISONYQ col iter n
```

displays the Nyquist curve, with col, iter and n as above. However, for this command it is to increase the number n by 1 for each new display, to ensure that the iteration number be properly displayed. (The modified Nyquist curve is displayed by means of the command SISOSTAB col iter n).

To obtain the final value of the design parameters use the command PRINT Z. To obtain intermediate values, use the command

case, an experienced designer can execute the algorithm one step at a time using the commands STEPn, n = 2,3,45,6, 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 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. A certain amount of problem reconditioning can be obtained by modifying the PUSH factors which may be found in the symbol table (see [B3]).

The designer may change the compensator, the design constraints and the algorithm parameters when control is returned to him. The package also includes an interrupt feature which can be used to interrupt macros such as RUN.

2.5 Analysis of Results

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

```
PRINT
```

can be used to dispaly any value in the 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

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 the value of the cost function; PSI gives 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 [G1]. All constraints are satisfied if PSI is zero.

The number E is monotonically reduced during the optimization. Sometimes the optimization becomes suspended because the quadratic program which computes the descent direction fails. When this happens a message appears. To "resuscitate" the program, it is necesary 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 difficulty due to a poor initial design or to poor problem scaling. In that

```
setamp = ?

#.05

trise = ?

#.5

tset = ?

#2.5

tfinal = ?

#5.

do you wish to use the default values e = .2, oldstp = 1.
?

#NO

type in value for e

#

type in value for oldstp

#100
```

2.4 Optimization Execution

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

#NO

type in z(1)

#1.

type in z(2)

#1.

type in z(3)

#1.

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

#YES do you wish to use the default gain margin = 2.2 and phase margin = 45?

#YES

do you wish io have constraints on the step response?

#YES

please give values for the above diagram

over = ?

#1.1

risamp = ?

#.7

in the modified version).

The program then asks for a description of the specifications in conversational mode, as shown below. The following responses are reasonable for the problem defined by the dialogues 1 and 2, above. The program defines the functions $g^j(z)$ from the inequalities: $bl(i) \leq z(i) \leq bu(i)$, $i = 1,2,3$. The user's responses are in upper case.

```
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.

do you wish to use the default values z(i) = bu(i)?
```

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 has a slight undershoot. 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 commands SISOSTEP and SISONYQ verify 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

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

~~s + 0.1s~~

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 10 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 shows that the overshoot is 15%. The Nyquist curve 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 is a simple,extendable, BASIC like language. It is a very small language which can be used to convert a set of FORTRAN subroutines into an interactive package. These subroutines are accessed via INTRAC commands. INTRAC has a

fairly powerful macro facility which makes it possible to write new commands in INTRAC itself rather than in FORTRAN. The commands written in FORTRAN are much more difficult to implement, but they execute much more rapidly. INTRAC has been used in a number of other packages as well, viz. SIMNON, IDPAC, MVDPAC, POLPAC AND SYNPAC, see [A1]. INTRAC has provisions for the control of execution of a FORTRAN program and for input and output. The output can be graphical.

INTRAC-C is an extension of INTRAC which includes a scratchpad for matrix calculations, elementary commands for color graphics, and commands for SISO design (see [B1,B3]). The commands for SISO design were constructed by dismembering the CDP package. INTEROPTDYN-SISO includes an extensive library of macros which combine the various elementary INTRAC commands into higher level commands. A complete list of elementary INTRAC and INTRAC-C commands is given in Appendix B. Some of the commands implemented as macros are discussed below.

4.2. Macros for optimization

The flow of the optimization process 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 both a number of 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 graphics. A full list of macros for graphics is given in Appendix D.

4.4. Macros for matrix calculations.

The matrix calculations facility in INTRAC-C induces a scratch pad with macro capability, but without internal variables. 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.

The macros for matrix calculations combine elementary commands for matrix calculations into very convenient commands which eliminate the need for declaring matrix dimensions. A full list of available macros is given in Appendix

E.

ACKNOWLEDGEMENT: This research was supported by the National Science Foundation under grants ECS-79-13148 and CEE-81-05790 and the Joint Services Electronics Program under grant F49620-79-C-0178. The programming at Berkeley was done by M. A. Bhatti, T. Essebo, E. Newman, W. Nye, E. Polak, and A. 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.

[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.

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, step 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.

nyquist: to plot nyquist diagram.

bode: to plot bode diagrams.

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

A.2. Macros for Optimization Execution.

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.

sisonyq c i k:	displays 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 begining with i th (the frequency 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.
parabola c; 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.

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/idsables echoeing 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 filehandldler 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 scratcpad

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. (does not have c).

HELP - Explains usage of the commands

LIST - Lists a macro file on the terminal or deposits it in the file fort.8.

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 specifed 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.

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 andis 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>
                        |<unasssigned 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.7. Macros for General Use

HLPEX - Prints a list of macros for execution.

HLPGR - Prints a list of macros for graphics

HLPPAD - Prints a list of macros which make use of scratch pad easier

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 infor- mation 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 calcuations 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 thru 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