Synpac Commands

User's Guide

Wieslander, Johan

1980

CODEN: LUTFD2/(TFRT-3159)/1-130/(1980)

Synpac Commands

-User´s Guide

JOHAN WIESLANDER

INSTITUTIONEN FÖR REGLERTEKNIK
LUNDS TEKNISKA HÖGSKOLA
1980

SYNPAC COMMANDS

–

User´s Guide

Johan Wieslander

Organization
LUND INSTITUTE OF TECHNOLOGY
Department of Automatic Control
Box 725
S-220 07 Lund 7 SWEDEN

Document name
Project report STU

Date of issue
February 1980

CODEN: LUTFD2/(TFRT-3159)/1-130/(1980)

Author(s)
Johan Wieslander

Sponsoring organization
Swedish Board for Technical Development (STU), contract No 77-3548

Title and subtitle

Synpac Commands - User's Guide

A4 A5

Abstract

Synpac is an interactive program, command oriented with a powerful macro facility. The program is aimed at design of feedback and feedforward controllers for multivariable linear dynamic systems. State reconstructors such as Kalman filters and Luenberger observers may be designed as well as output feedback designed from a state feedback computed e.g. using a linear quadratic method.

The designed control system may be analysed either in the frequency domain using Bode diagrams, Nyquist diagrams, or Nichols diagrams, or in the time domain using simulation or eigenvalue plots.

Key words A4 A5

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title

ISBN

Recipient's notes

Number of pages
130

Price

Security classification

Distribution by (name and address)

Synpac Commands - User's Guide

This is the guide to the commands in the program Synpac. It will give information on how to call the different commands, what their function are and what method is used. In many cases hints and examples are incleded.

Synpac is an interactive command oriented program package. It is aimed at the design of control algorithms for multi input - multi output dynamical systems on state space form. Both continuous time and discrete time systems can be handled.

The intended design criteria are the shape and speed of time responses as well as pole/zero configurations and frequency responses. Also the influence of noise on the system may be a design criterion. The main but not only design tool is linear quadratic methods, used both for regulator and observer design. The 'design knobs' are in these cases the assignment of values for the respective quadratic criteria. This design scheme has proved intuitively natural and is easy to learn. It is supported with a function that helps in transforming the criteria matrices to a suitable form. Methods also exist to adapt the resulting design to a suboptimal, but maybe more realistic one.

To achieve these goals, Synpac contains the following facilities:

a) Matrix definition, handling and operations.
b) System definition.
c) Eigenvalue (pole) computations.
d) Frequency response evaluation.
e) Simulation.
f) Graphic output of time and frequency responses.
g) Solution of the stationary Riccati equation for optimal regulators and observers.
h) Design of reduced order regulators and observers.
i) Generation of deterministic and stochastic test signals.

Synpac is a member of a family of program packages with identical structures and interaction. Further reading on the interaction and the general use of these programs can be found in the two reports:

1. J. Wieslander, H. Elmqvist: INTRAC - A Communication Module for Interactive Programs - Language Manual, TFRT-7132, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

2. J. Wieslander: Interactive Prgrams for Analysis and Design of Control Systems - General Guide, TFRT-3156, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

These programs were developed with financial support from the Swedish Board of Technical Development, contracts 73-3553, 75-3776 and 77-3548. They represent the combined effort of many members of the department over several years.

Commands Available in Synpac

The following is a structured list of commands available in Synpac, together with a short indication of their use.

1. Input & Output

AGR	Edit an aggregate file
EDIT	Edit a symbolic file
LIST	Output data on user readable form
MOVE	Move data in the data base

2. Graphic Output

BODE	Draw curves in a diagram with logarithmic scales
HCOPY	Take a hard copy of the last graphic output
NIC	Display a frequency response in a Nichols diagram
NYQ	Display a frequency response in a Nyquist diagram
PLEV	Display eigenvalues etc in the complex plane
PLOT	Draw curves in a diagram with linear scales

3. Time Series Operations

CONC	Concatenate two time series
CORNO	Generate a correlated noise time series
CUT	Extract a part of a time series
INSI	Generate time series
PICK	Pick equidistant time points
SCLOP	Do scalar operations on a time series
STAT	Compute some statistical numbers
VECOP	Do vector operations on a time series

4. Matrix Operations

ALTER	Alter elements in a matrix
EIGEN	Compute the eigenvalues of a matrix
ENTER	Enter a matrix element by element
EXPAN	Generate a matrix from sub-matrices
MATOP	Perform matrix operations
REDUC	Extract a submatrix
UNITM	Generate a unit matrix
ZEROM	Generate a zero matrix

5. System Conversion & Analysis

CONT	Convert to continuous time form
POLES	Compute the poles of a system
SAMP	Convert to discrete time form
SIMU	Simulate the time response of a system
SPSS	Compute the frequency response of a system

SYSOP	- Generate a system from its subsystems
SYST	- Generate a system description
TRANS	- Convert a criterion from continuous time to discrete time form

6. Design

FEEDF	- Design feedforward control
KALFI	- Compute a Kalman filter gain
LUEN	- Help construct a reduced order observer
OPTFB	- Compute a linear quadratic state feedback
PENLT	- Reduce a penalty function to standard form
PPLAC	- Pole placement for single input systems
RECON	- State reconstruction for single input systems
REDFB	- Compute an output feedback

7. Miscellaneous

DELET	- Delet a file from the data base
FHEAD	- Inspect and change file parameters
FTEST	- Test the existence of a file
TURN	- Change program switches

8. Alphabetical Command List

AGR

Purpose

To create or to edit an aggregate file.

Command

```
AGR AGROUT
   or
AGR [AGROUT] < AGRIN

AGROUT   - name of resulting aggregate file
           by default AGROUT = AGRIN
AGRIN    - name of original aggregate file
```

Subcommands

The sub-commands implicitly use a pointer to the current component file.

LOOK [NAME]
Display the table of contents of AGROUT.
If NAME is present, then only the entries named NAME, if any, will be displayed. The pointer is not affected.
For each entry, one may see if it is flagged for insertion, deletion, and/or isolation.

KILL
Leave sub-command mode. Current AGR-command including sub-commands will have no effect.

X
Leave sub-command mode. Current AGR-command including sub-commands will take effect.

LOC NAME
Make the pointer point at the component file NAME. The scan takes place between the current pointer location plus one and the last entry, at which place the pointer remains in case of no success.

REP [NAME]
Replace the component file at the current pointer location by the individual file NAME.
The pointer is not affected.
By default NAME equals the name of the component file at the current pointer location.

DEL
Delete the component file at the current pointer location, then auto-decrement the pointer.

ISO
Short for isolate, i.e. copy the component file at the current pointer location to an individual file with the same name. The pointer is not affected.

TOP
Make the pointer point above the first component file.

INS NAME
Insert the individual file NAME after the current pointer location, then auto-increment the pointer.

BOT
Make the pointer point at the last component file.

REM
Remove the REP, DEL and ISO flags from the current pointer location. The pointer is not affected.

ADV [NR]
Advance the pointer NR steps. By default NR = 1. In case of no success, the pointer will remain at the last entry for a positive NR and at location 0 for a negative NR.

Function

The main command specifies the type of operation, i.e. update or generate an aggregate file. The component files are manipulated by subcommands in a way similar to that of a line oriented text editor. Note that no I/O operations (other than file existence tests) take place until the excute subcommand (X) is entered; the specified operations are only entered into a table. This table may be viewed through the subcommand LOOK, and errors may be corrected. At the time of execution, the operations are performed from the top of the table, thus the chronological order in which the operations were entered is immaterial.

EDIT

Purpose

To edit, i.e. create or make changes to a symbolic (text) file. Examples are MACRO-files, system files and symbolic data files from outside.

Command

EDIT TFILE

TFILE name of symbolic (text) file

Subcommands

The following general notation is used:

n denotes a positive integer, default 1.

/ denotes any character not included in 'string'.

string denotes any sequence of printing characters including space.

A string the string is appended to the current line.

B the bottom line of the file is made the new current line.

C /string1/string2/ string1 in the current line is changed to string2.

D [n] n lines are deleted starting with the current line.

E exit, i.e. close the file and return.

F string find the first line after the current line starting with string and make it current.

I string insert string as the new current line after the now current line.

L string locate the first line after the current line containing string and make it current.

N [n] make the n:th next line current.

O [n] — overlay the n next lines including the current with keyboard INPUT.

P [n] — print n lines starting with the current line. The last line printed is the new current line.

R string — replace the current line with string.

T — go to the top of the file.

DIS ON
DIS OFF — enable/disable output on display.

Function

The editor works in one of two modes, EDIT-mode and INPUT-mode. In EDIT-mode, the editor will read the text-file line by line. At any time, one line is the 'current line'. The subcommands control the position of the 'current line' within the text-file, or modify the 'current line'.

In INPUT-mode, a line typed on the keyboard is made the new 'current line', thus forcing the old one to be written to the output file.

The initial mode of the editor is INPUT if the specified file is not found, otherwise EDIT. An empty line is used to switch the mode.

Cautions, Restrictions

EDIT does not allow subcommands or input from a macro.

Hints

System files are normally generated by the command SYST. Exception: the polynomial image system files in Idpac.

LIST

Purpose

To output a file on lineprinter, teleprinter, or display.

Command

```
LIST [(DEV)][(FEED)][(DMODE)][AGGREG:]FNAME[(A1 A2..)][IF NUM]

DEV     - device = 'DIS'/'LP'/'TP'
          DIS    - display
          LP     - line printer
          TP     - teleprinter
          (by default DEV = 'DIS')
FEED    - form feed parameter = 'FF'/'LF'
          FF     - a form feed will precede output
          LF     - a line feed will precede output
          (by default FEED = 'LF')
DMODE   - data mode indicator = 'D'/'T'/'DS'/'TS'/'FT'/'FTS'
          D      - FNAME is assumed to contain binary data
          T      - FNAME is assumed to contain text
          DS     - same as 'D', but sequence numbers written
          TS     - same as 'T', but sequence numbers written
                   and the text will be truncated after
                   72 characters
          FT     - same as 'T' but 'BEGIN', 'END' not written
                   provided that section names
                   have been given explicitly
          FTS    - same as 'FT' with sequence numbers
          (by default DMODE = 'D')
AGGREG  - aggregate file, invalid in connection with
          DMODE = 'T'
FNAME   - name of file to be listed
A..     - attributes associated with FNAME, if
          DMODE = 'D'/'DS', then A.. denotes column numbers,
          otherwise names of sections within FNAME
IF      - number of 1st record to be listed
          (valid only in connection with DMODE = 'D'/'DS')
NUM     - number of records to be output
          (valid only in connection with DMODE = 'D'/'DS')
```

Function

The data is printed as matrix blocks with NUM lines containing the first few columns, a blank line, NUM lines containing the next few columns etc.

Note, frequency response files are special cases of data files.

Text files: The file is directly copied onto the output medium. Text files are:

a) any file created or manipulated by the EDIT command,
b) MACRO files,
c) system files,
d) structure files.

If a section name is given for a system file, only that section is output.

Cautions, Restrictions

The available output devices may be installation dependent.

Hints

The mechanism that allows listing of selected sections of a system file uses the keywords BEGIN and END. These keywords may be used to produce the same effect in any text file, e.g. to output descriptive text from a macro.

Examples

```
>LIST DATA
>LIST (LP) DATA(3 4 6) 20 10
>LIST (T) MAC
>LIST (LP)(T) SYST(NAME)
```

MOVE

Purpose

To move files in the data base. The columns of a data file may be rearranged.

Command

```
MOVE [(DMODE)] [[AGOUT:]FILOUT[(C11..)]] <
                                  [AGIN:]FILIN[(C21..)]

DMODE    - data mode = 'D'/'T'/'ND' (default: DMODE = 'D')
           D      - the file is assumed to contain binary data
           T      - the file is assumed to contain text
           ND     - same as 'D' but the columns C11.. will,
                    if previously defined, not be overwritten,
                    but placed rightmost in FILOUT, in the
AGOUT    - output aggregate file
FILOUT..- output file name [with column numbers]
AGIN     - input aggregate file
FILIN.. - input file name [with column numbers]
```

Function

The columns C21,.. in the data file FILIN are moved to the columns C11,.. in the data file FILOUT.

Copying is the only function available for symbolic (text) files or for files within aggregates.

Cautions, Restrictions

- Column numbers cannot be used for system- and macro-files.
- Data files may contain up to 20 columns as input files and up to 15 columns as output files.

Examples

```
>MOVE WORK < DATA(2 5 3)
>MOVE (ND) WORK(1 3) < DK DATA(4 1)
```

The results are shown below.

```
Input & Output
MOVE


DATA

    11.0000    21.0000    31.0000    41.0000    51.0000
    12.0000    22.0000    32.0000    42.0000    52.0000
    13.0000    23.0000    33.0000    43.0000    53.0000
    14.0000    24.0000    34.0000    44.0000    54.0000
    15.0000    25.0000    35.0000    45.0000    55.0000
    16.0000    26.0000    36.0000    46.0000    56.0000
    17.0000    27.0000    37.0000    47.0000    57.0000
    18.0000    28.0000    38.0000    48.0000    58.0000
    19.0000    29.0000    39.0000    49.0000    59.0000
    20.0000    30.0000    40.0000    50.0000    60.0000


WORK

    21.0000    51.0000    31.0000
    22.0000    52.0000    32.0000
    23.0000    53.0000    33.0000
    24.0000    54.0000    34.0000
    25.0000    55.0000    35.0000
    26.0000    56.0000    36.0000
    27.0000    57.0000    37.0000
    28.0000    58.0000    38.0000
    29.0000    59.0000    39.0000
    30.0000    60.0000    40.0000


WORK

    41.0000    21.0000    11.0000    51.0000    31.0000
    42.0000    22.0000    12.0000    52.0000    32.0000
    43.0000    23.0000    13.0000    53.0000    33.0000
    44.0000    24.0000    14.0000    54.0000    34.0000
    45.0000    25.0000    15.0000    55.0000    35.0000
    46.0000    26.0000    16.0000    56.0000    36.0000
    47.0000    27.0000    17.0000    57.0000    37.0000
    48.0000    28.0000    18.0000    58.0000    38.0000
    49.0000    29.0000    19.0000    59.0000    39.0000
    50.0000    30.0000    20.0000    60.0000    40.0000
```

BODE

Purpose

To plot frequency response files in Bode diagram format.

Command

```
BODE [(SW)] FRF1[(F11 F12 ..)] [FRF2[(F21 .. )] ..

SW       - page switch = 'A'/'P'/'AP'/'AO'
           (default: 'AP')
           A : plot amplitude only,
               then read a sub-command
           P : plot phase only, then exit
           AP: plot amplitude and phase together,
               then exit
           AO: plot amplitude only, then exit

FRF..    - frequency response file name(s)
F11..    - frequency response number(s) (default all)
```

Subcommands

```
PAGE     - request the phase plot (relevant only if SW = 'A')
KILL     - skips the phase plot (relevant only if amplitude
           and phase are to be plotted separately)
```

Function

The indicated (default all) curves of the frequency response file(s) FRF1 etc., are plotted versus frequency. The abscissa is a logarithmic axis, while the ordinate is logarithmic for the amplitude and linear for the phase. If the phase information is identically zero, as for auto spectra, the phase plot is omitted.

Amplitude values smaller than 1.E-5*(largest value) are replaced by the lower limit.

If more than one set of curves are requested, they are marked with integers representing the order of the corresponding response in the command.

Hints

a) Cf. the command PLOT for methods to include text in the diagram.

b) The command treats frequency response files, see the general guide. Generally, they include frequency information scaled in rad/s. If you want a bode plot in Hz, use the SCLOP command to divide the frequency by 2π = 6.2831853.

<u>Example</u> (cf. the commands NIC and NYQ)

The Bode plot for the system

$$G_O = \frac{3.25\ s + 1}{s^2\ (s + 1.75)}$$

is given through the command (the response is contained in the file FRF):

```
>BODE FRF
```

See Figure 1.

The Bode plot for the system

$$G_c = \frac{3.25\ s + 1}{s^3 + 1.75\ s^2 + 3.25\ s + 1}$$

is shown in Figure 2 (the response was contained in FRF2).

Graphic Output
BODE

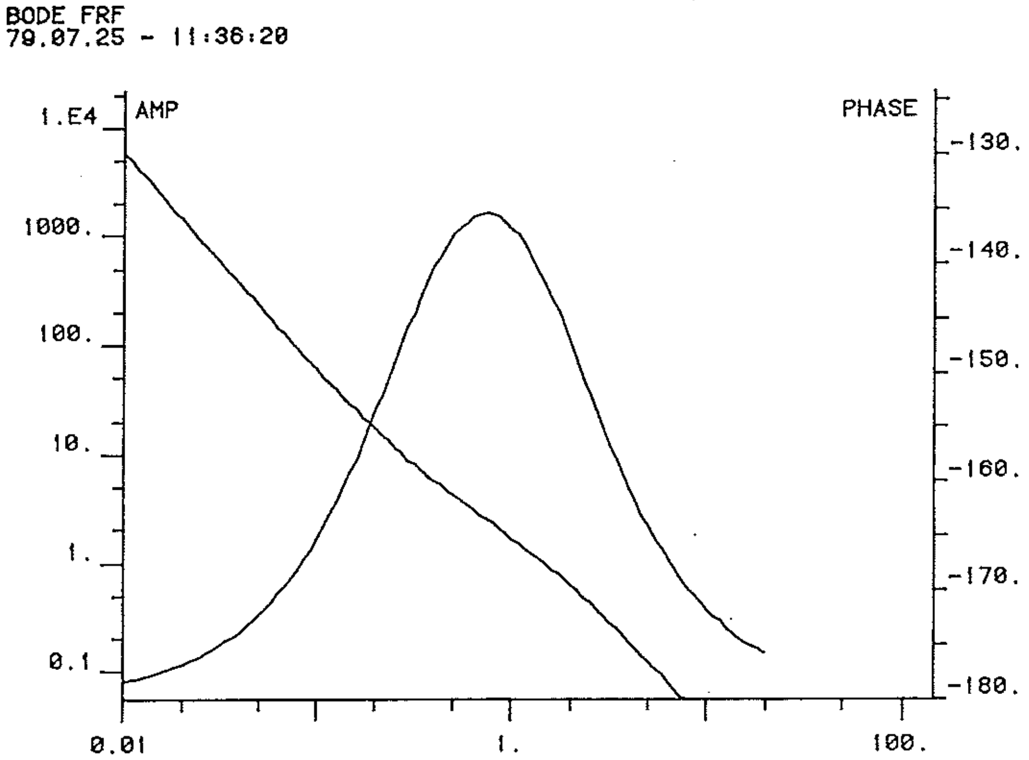


Figure_1. The amplitude and phase of G_0.

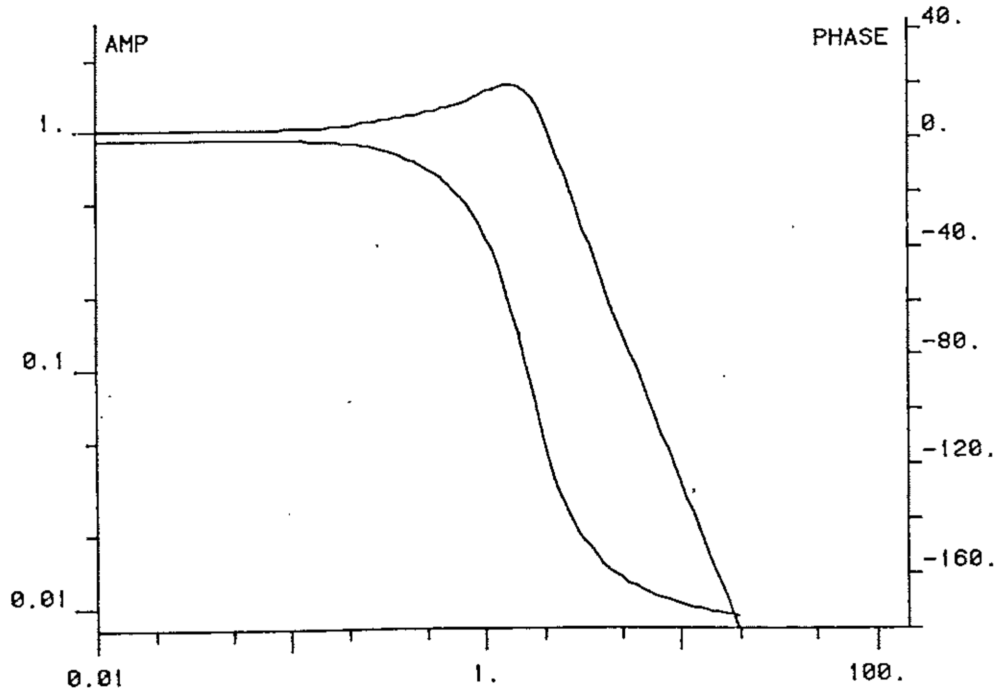


Figure_2. The amplitude and phase of G_c.

HCOPY

Purpose

To generate a hard copy of graphical output.

Command

```
HCOPY [DEV] [FACTOR]
   or
HCOPY SWITCH

DEV      - hardcopy device = 'L'/'R' (default: 'L')
           L: local hardcopy (Tektronix 4662)
           R: remote hardcopy (Calcomp 1051)
FACTOR   - scale factor (default: 1.)
           For L: .5 < FACTOR < 1.6
           For R: .5 < FACTOR < 4.
SWITCH   - = 'ON'/'OFF'/'T'
           ON : enables hardcopy
           OFF: disables hardcopy
           T  : output a free text string at the current
                joystick location,applies to TEKTRONIX 4662
                only, the textstring in the command line
                being preceded by a double quote
```

Function

After that the command HCOPY has been used with the switch ON, all graphical output that is generated in any command is also saved temporarily. A subsequent use of the command HCOPY will cause the saved information from the last such command to be sent to the selected hard copy device.

Hints

Note that HCOPY actually is available as a subcommand for all graphic generating command.

NIC

Purpose

To plot frequency response files in a Nichols diagram.

Command

```
NIC [WMIN WMAX] FRF1[F11 ..)] [FRF2...]

WMIN, WMAX- frequency limits, default: all frequencies
            plotted
FRF..     - frequency response filename(s)
F11..     - frequency response number(s), default: all

Note: max 5 curves may be displayed
```

Function

The indicated (default all) curves of the frequency file(s) FRF1 etc., are plotted in a rectilinear coordinate system, where the horizontal axis is linear and represents the phase in degrees and the vertical represents the magnitude and is logarithmic. Frequency points of the form $1*10^n$, $2*10^n$, and $5*10^n$ (n integer) are indicated on the curve(s).

If a plotted curve is the frequency response of an open loop transfer function G_0, the corresponding closed loop transfer function $G_c = G_0/1+G_0$ can be read from the curvilinear coordinate system. These curves represent the magnitude of the closed loop transfer function in dB and its phase in degrees.

If more than one curve are requested, they are marked with integers according to the order of the response in the command.

Cautions, Restrictions

There can be no more than five curves in a single diagram. Often only a part of a frequency response will fit into the diagram. Then the optional arguments WMIN and WMAX are recommended.

Hints

Cf. the commands BODE and NYQ.

Example

(Cf. the commands BODE and NYQ.) The frequency response for $0.2 \leq \omega \leq 5$ is shown in a Nichols diagram:

```
>NIC 0.2 5 FRF
```

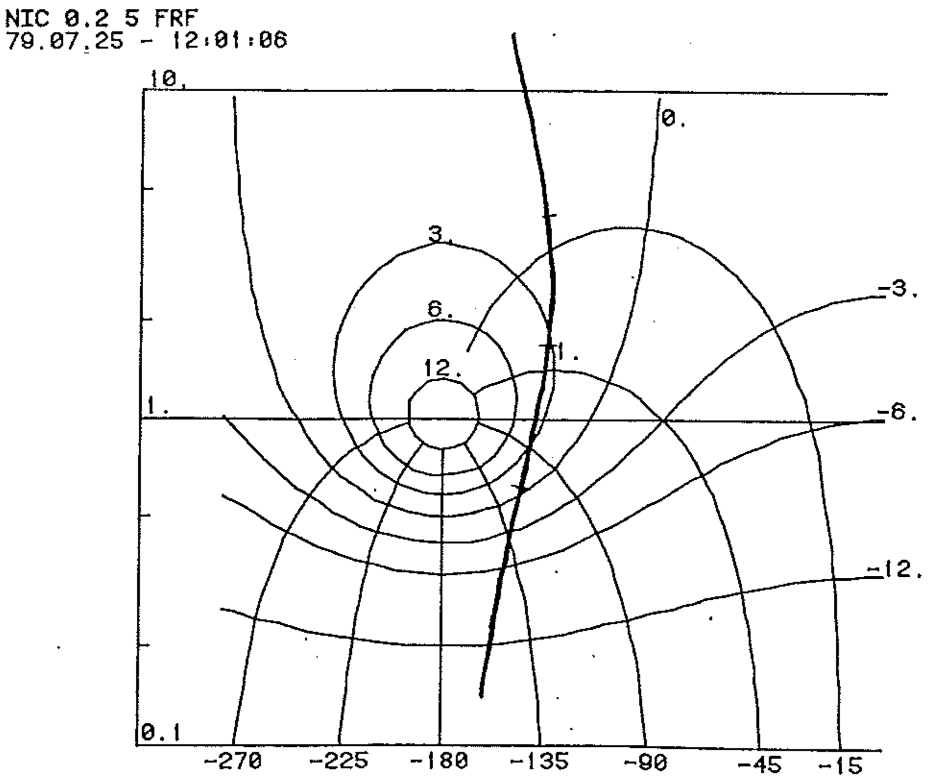


Figure 1

NYQ

Purpose

To plot frequency response files in a Nyquist diagram.

Command

```
NYQ [WMIN WMAX] FRF1[(F11 ..)] [FRF2 ...]

WMIN, WMAX - frequency limits, default: all frequencies
             plotted
FRF..      - frequency response filename(s)
F11..      - frequency response number(s), default: all
```

Function

The indicated (default all) frequency responses are plotted. For each frequency its associated value, being a complex number in polar form, is marked in the complex plane and a line is drawn to the preceding point, so forming a Nyquist curve.

Frequency points of the form $1*10^n$, $2*10^n$, and $5*10^n$ (n integer) are indicated on the curve(s).

Cautions, Restrictions

No more than five curves can be plotted in a single diagram. It may be essential to use the arguments WMIN and WMAX in order to obtain a reasonable scale on the axes.

Hints

Cf. the commands BODE and NIC.

Example

(Cf. the commands BODE and NIC.) The Nyquist plot of the open loop system

$$G_0 = \frac{3.25\ s + 1}{s^2\ (s + 1.75)}$$

looks like (see Figure 1):

Graphic Output
NYQ

```
>NYQ 0.5 10 FRF
```

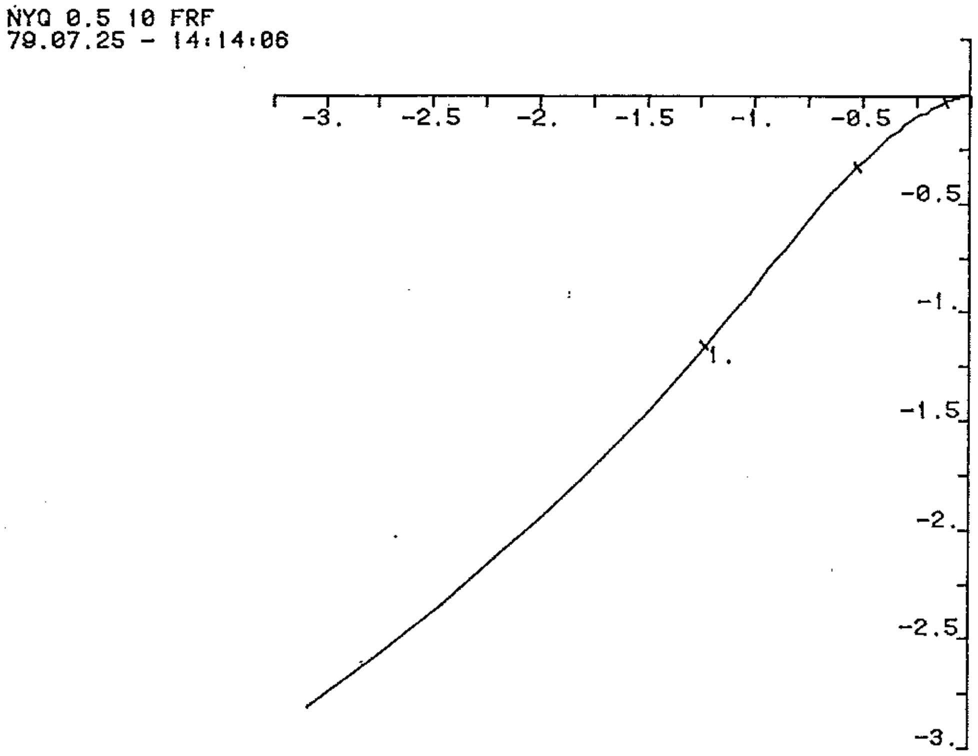
The corresponding closed loop system

$$G_c = \frac{3.25\ s + 1}{s^3 + 1.75\ s^2 + 3.25\ s + 1}$$

looks like (its response is in FRF2):

```
>NYQ 0.2 5 FRF2
```

See Figure 2.



NYQ 0.5 10 FRF
79.07.25 - 14:14:06

<u>Figure 1</u>. The Nyquist plot of G_0.

Graphic Output
NYQ

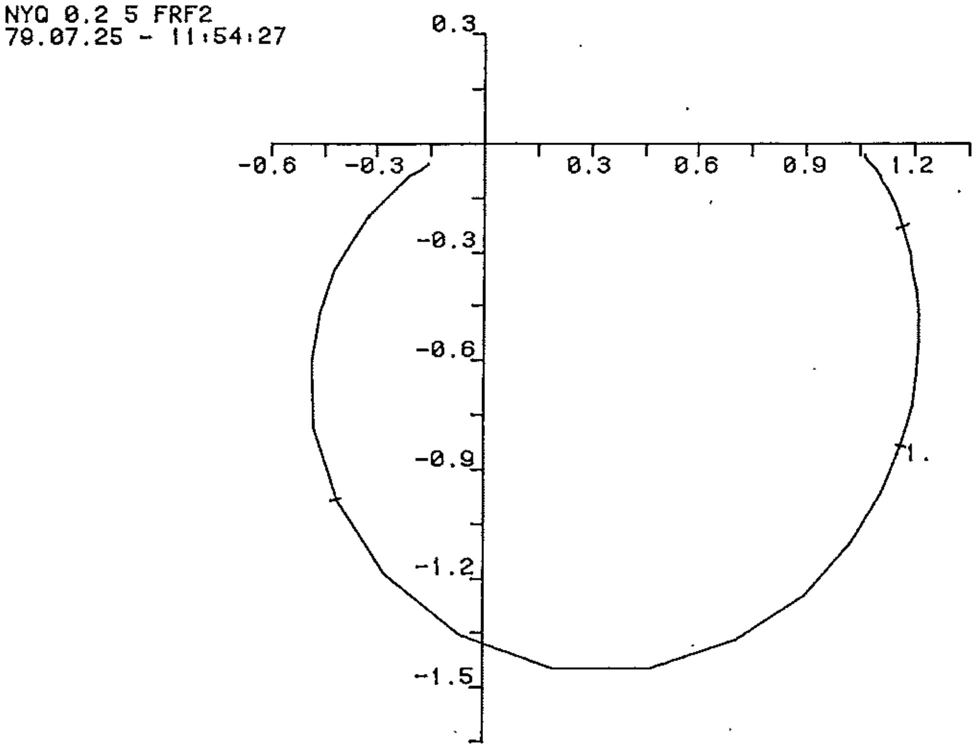


Figure 2. The Nyquist plot of G_c.

PLEV

Purpose

To plot and optionally edit the contents of a locus file, e.g. eigenvalues, poles and zeros.

Command

```
PLEV FNAM2 [FNAM3 FNAM4 ...]
  or
PLEV  FNAM1 < FNAM2

FNAM1             - locus file containing new eigenvalues
FNAM2,FNAM3 ... - locus filescontaining original eigenvalues
```

Subcommands

ALT N VR [VI]
Alter eigenvalue number N: real part to VR and imaginary part to VI (default = 0.). If the former imaginary part of eigenvalue number N is not equal to zero, the complex conjugate is also altered, to VR and -VI. It is not allowed to alter a single real eigenvalue to a complex. In that case, use the second form:

ALT N1 VR VI & N2
Alter the two real eigenvalues number N1 and N2 to complex conjugated eigenvalues. The real parts are altered to VR and the imaginary parts to +VI and -VI respectively.

SCALE N V
Scale eigenvalue number N and its omplex conjugate, if there is any, by the scaling-factor V.

DAMP N Z
Move eigenvalue number N and its complex conjugate along a circle to achieve the relative damping Z, 0 < Z < 1.

EXAM N
Write the real and imaginary parts of eigenvalue number N. If the imaginary part is not equal to zero, the relative damping, the distance to the origin and the angle to the negative real axis are also written.

LOOK
Plot eigenvalues on display and write the numerical values if there is room for them on the screen. The eigenvalues are ordered after decreasing values of the real parts.

X
   Ends the cubcommand sequence and generates output.

KILL
   Aborts the subcommand sequence.

Function

The information is plotted in a complex plane shown on the terminal. A unit circle is included for discrete time locus files. The command exists in two forms. The first form serves to visualize the information from one or more locus files but does not allow changing it.

The second form allows only one locus file as input, but generates a new one as output with changes made through subcommands.

The operations through the subcommands automatically change complex conjugates and a special form exists to change two reals to a complex conjugate pair. The eigenvalues are numbered in order of decreasing real parts. They are also listed in this order on the terminal (if TEXT is ON).

To identify different sets of eigenvalues (several original sets or an old and a new one), the following sequence of symbols are used: X, □, ∇, , 5, 6,... .

Examples

Assume that the locus file LOCUS is given, containing 3 eigenvalues all equal to zero. The command

```
     >PLEV LOCUSN < LOCUS
```

will display the original values, see Figure 1. The subommands

```
     >ALT  1  -1
     >ALT  2  -0.5  0.5  &  3
     >LOOK
```

will then give the output shown in Figure 2.

```
     >SCALE  1  1.5
     >LOOK
```

then results in Figure 3. Finally the relative damping is specified to 0.9 giving Figure 4:

```
     >DAMP  1  0.9
     >LOOK
     >X
```

Graphic Output
PLEV

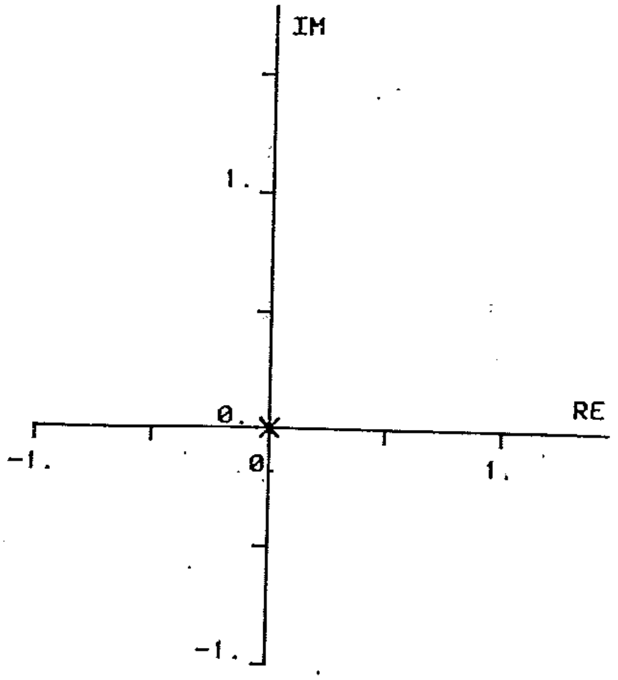


Figure_1. The original set of eigenvalues.

Graphic Output
PLEV

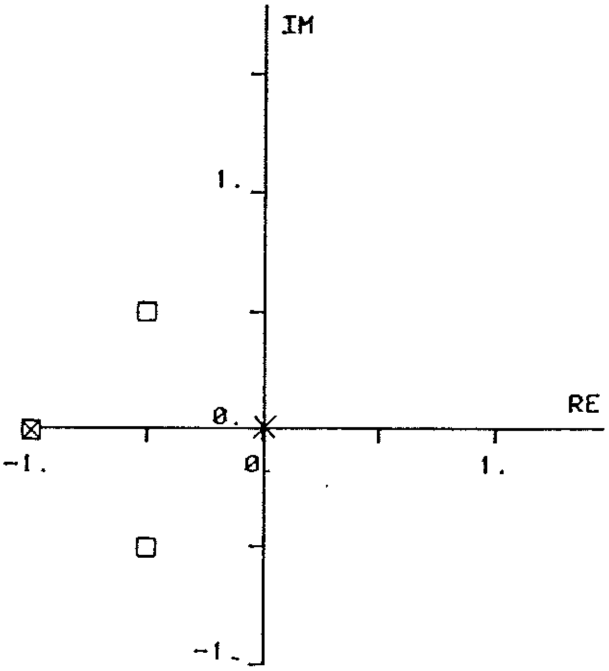


Figure_2. The eigenvalues after the first change.

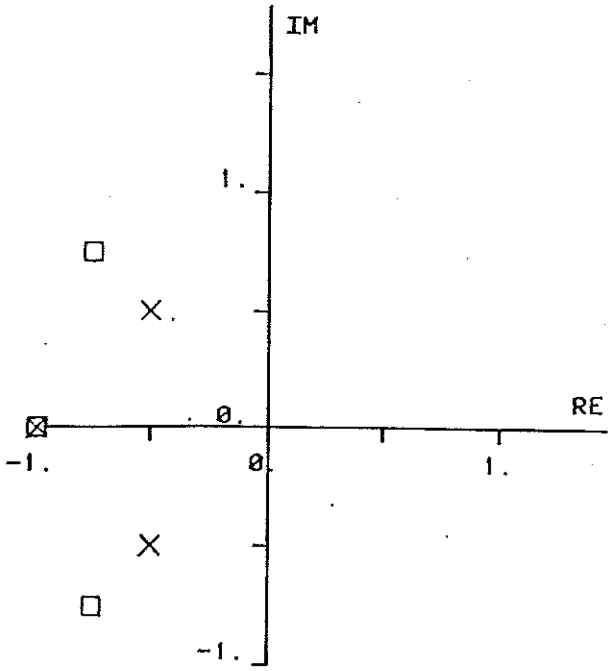


Figure_3. Moving a complex pair with constant damping.

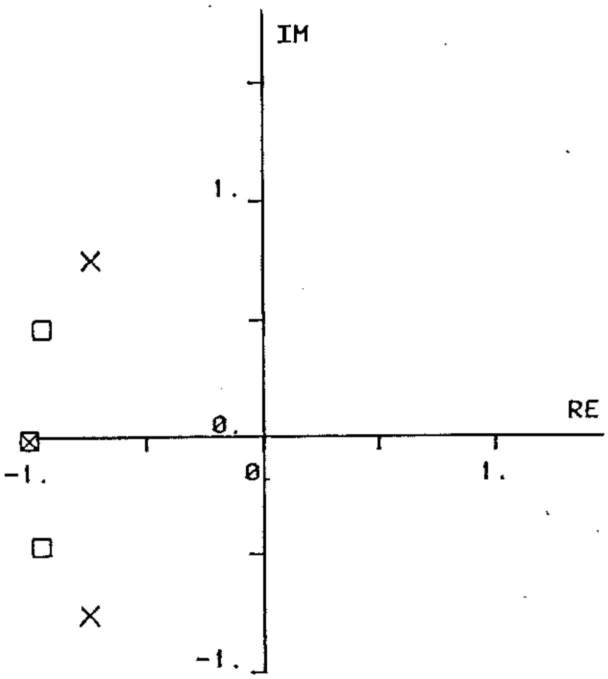


Figure 4. The final eigenvalue positions.

PLOT

Purpose

To show data in diagrams with linear scales.

Command

```
PLOT [(M)] [(NP)] [FNAMX[(C1..)] < ] [(OPT1)] FNAM1[(C11..)]
     [[(OPT2)] [FNAM2[(C21..)]] .. ] [YMI YMA]
```

```
M       - mark option = 'M'/'NM' (default: 'M')
          M      - when more than one curve is plotted,
                   the curves are marked with integers
                   representing the order of the
                   corresponding column in the PLOT command
          NM     - no curve marks
NP      - nr of points per page (by default NP=NPLX.)
FNAMX   - optional file containing x-values if plotting
          versus time/sample number not wanted
C1      - column number for FNAMX (by default C1=1)
OPTi    - plot option='LI'/'HP'/'NL' (by default OPT='LI')
          LI: linear interpolation plot
          HP: histogram plot
          NL: no lines will connect the plotted points
FNAMi   - i:th data file containing the y-values
Cji..   - column numbers of FNAMj
YMI     - minimum value for this plot (by default YMI=YMIN.)
YMA     - maximum value for this plot (by default YMA=YMAX.)
```

Screen split operators inserted between groups of file names:

```
'/'     - divides the plotting area horizontally
':'     - divides the plotting area vertically
```

These operators express the mutual positioning of graph groups. It is allowed to divide the screen at most into three parts horizontally and two parts vertically.

Subcommands

```
KILL    - skip all the following plot pages
PAGE    - plot the next plot page in turn
SKIP [N]- skip the N (deafault: 1) following plot pages
```

PLOT

Function

The indicated column(s) in the data file(s) are shown in graphic form, normally directly on the operators terminal.

In normal operation, the information is shown along the vertical axis versus a linearly increasing variable on the horizontal axis (time). The scaling of the time variable may, by use of the TIME switch be altered from sample number in the command TURN, to hour, minute or second. If the argument FNAMX is included, the variable shown along the horizontal axis is taken from a file with this name. The x-axis is marked with an H, M or S depending on the time unit, with NR if the sample number is used, otherwise it is left blank.

The number of values shown on a single page is determined from the argument NP if present, else from the global variable NPLX. The global variables referenced by the PLOT command are given default values at program start up. They may be inspected by the WRITE statement of Intrac and may be modified with the LET statement. The TIME switch determines the interpretation of NP (NPLX.).

When more than one variable is shown, they are identified by integers according to the order of the variables in the command line.

The scaling along the vertical axis is determined in the following way:

a) The arguments YMI and YMA are present or the global variables YMAX. and YMIN. satisfies YMAX.>YMIN.: Then YMI (YMIN.) is used as the minimum value on the axis and YMA (YMAX.) is used as the maximum value.

b) Otherwise automatic (i.e. data dependent) scaling is performed. Automatic scaling is always used if the screen-split operations are used.

Automatic scaling is influenced by the global variable SCALES.:

If SCALES.≠0 then scale marks are multiples of either 1.0, 1.5, 2.0, 2.5, or 5.0.

If SCALES.=0 then scale marks are multiples of either 1.0, 2.0, or 5.0.

Hints

a) Note that the command line is written on the display above the plot and that comments can be added (after a double quote).

b) If the users terminal is a display with a direct hard copy option, the command WRITE (DIS) ... may be used to add text to the plot.

c) The command HCOPY may be used to include text on an on-line plotter if such is available.

Examples

V1 and V2 are two given signals. They may be visualized in the following way:

```
>PLOT V1 V2
```

giving the result shown in Figure 1. Alternatively, they may be shown in the following way:

```
>PLOT V1 / V2
```

The result is shown in Figure 2.

In order to visually study their interrelation, a scatter diagram might be constructed using an x-y plot and the "No Lines" switch:

```
PLOT V1 < (NL) V2
```

It is shown in Figure 3.

A pulse with IFP.=3 and LENGTH=2 (cf. the command INSI) looks like this (Figure 4) with and without the "HP" switch:

```
>LET IFP.=3
>INSI PULSE 7
    >PULSE 2
    >X
>PLOT (7) PULSE / (HP) PULSE
```

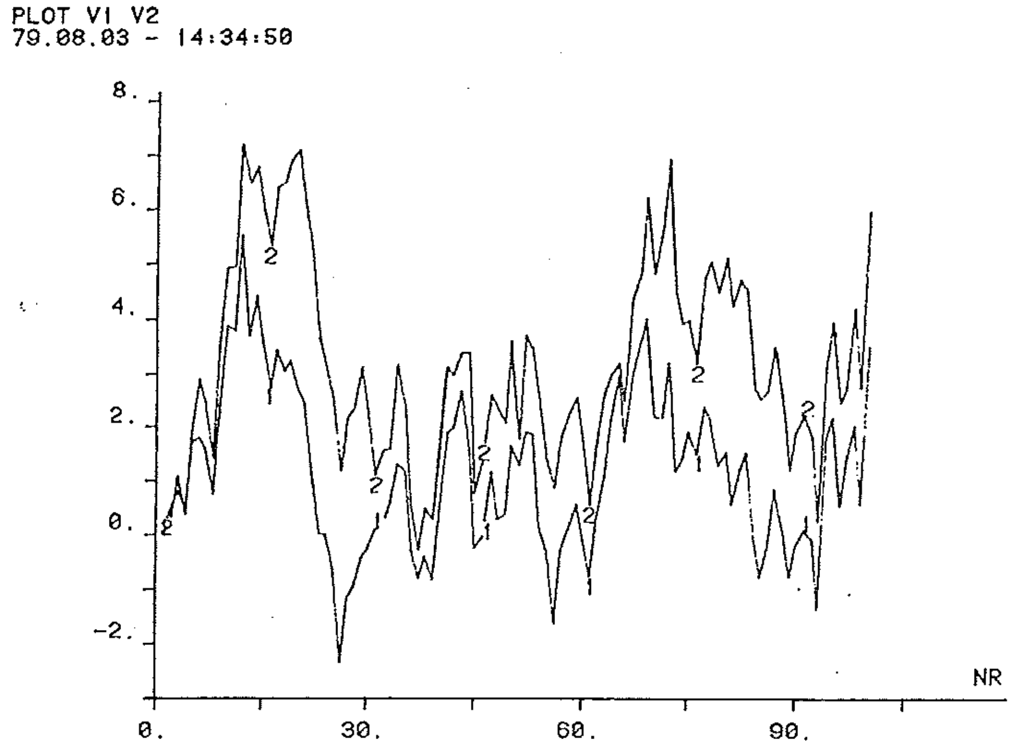


Figure_1. A plot with two curves in the same diagram.

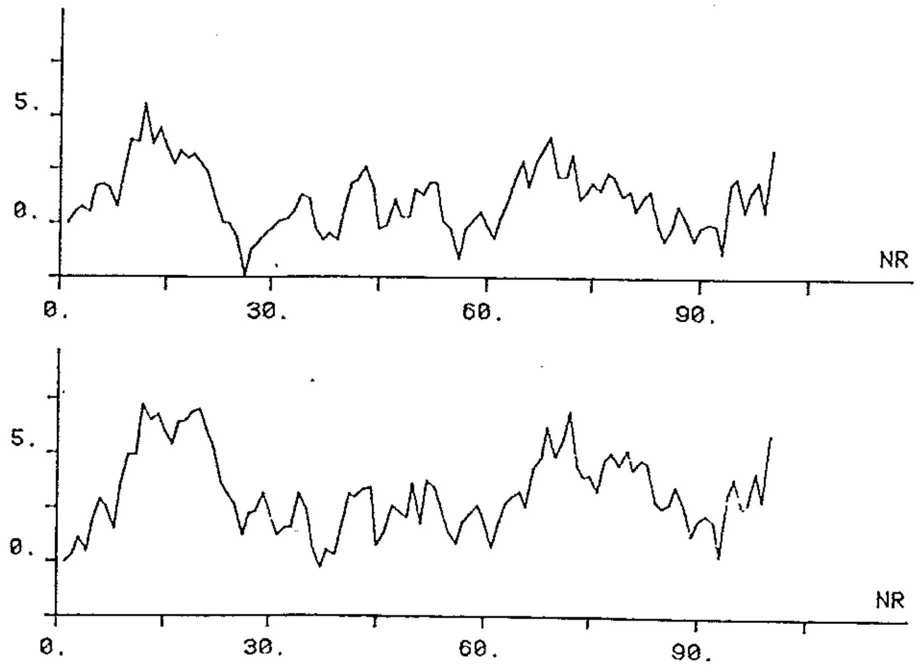


Figure_2. A plot with the screen split horizontally.

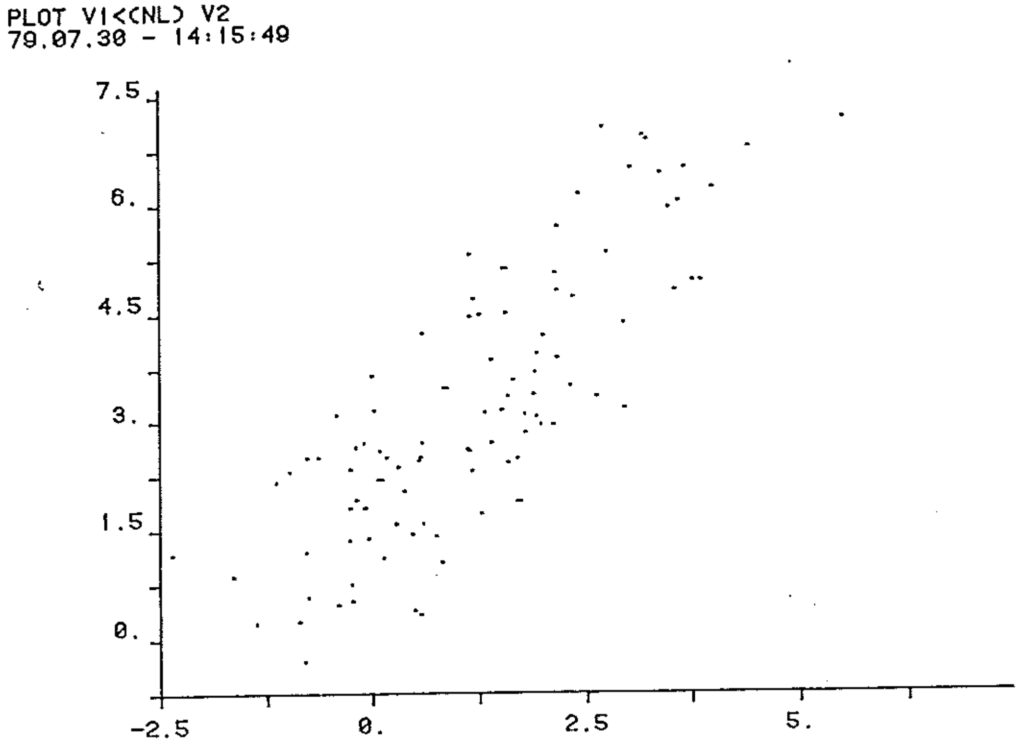


Figure 3. A plot in the x-y mode of operation.

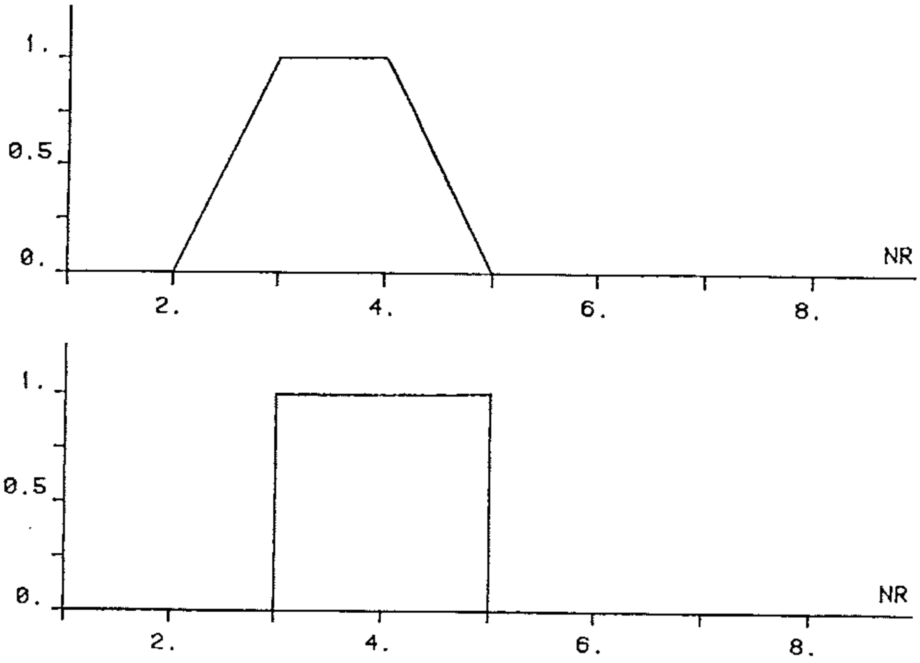


Figure 4. An illustration of linear interpolation vs. histrogram plot.

CONC

Purpose

To concatenate two time series files.

Command

```
CONC [FNAM1] < FNAM2 FNAM3

FNAM     - names of output and input files, resp.
           (by default FNAM1 = FNAM2)
```

Function

The files FNAM2 and FNAM3 are concatenated giving the file FNAM1. They must contain the same number of columns.

CORNO

Purpose

To generate a time series of correlated gaussian noise vectors. The noise is uncorrelated in time but with a given covariance matrix between the noise components.

Command

```
CORNO NOISE < [AG:]COMAT NSAMP
   or
CORNO V E < SYST[(NAME)] NSAMP
```

NOISE	- noise vector file name
AG	- aggregate file name
COMAT	- covariance matrix file name
V	- stochastic input signal file name
E	- measurement error signal file name
SYST	- name of system file containing a covariance function
NAME	- name of section within SYST
NSAMP	- number of samples wanted

Function

In the first form of the command, the noise covariance matrix is given explicitly . In the second form, the V and E stochastic inputs (cf. Fig. 4.5 in the General Guide), are generated from the matrices R1, R12, and R2 given in the system description SYST(NAME).

The command uses and alters the global variable NU. which is the state of the random number generator.

Method

The covariance matrix R for the derived noise vector is either given directly in the comand line (form 1) or through

$$R = \begin{bmatrix} R_1 & R_{12} \\ R_{21} & R_2 \end{bmatrix} \qquad R_{21} = R_{12}^T$$

where R_1, R_{12}, and R_2 are given in the system description (form 2).

R is then factored as

$$R = S \cdot S^T.$$

The noise vector n is then computed from the uncorrelated noise vector e through

$$n = S^T \cdot e.$$

The last step is repeated NSAMP times giving the required time series. If R was given in the blocked form (form 2), V and E are obtained through a similar partitioning of n.

Cautions, Restrictions

The covariance matrix must be symmetric, positive semidefinite.

CUT

Purpose

To cut out a part of a time series file.

Command

```
CUT [FNAM1] < FNAM2 IB NUM

FNAM1   - output file (default: FNAM2)
FNAM2   - input file
IB      - first record to be saved
NUM     - number of records to be saved
```

Function

The rows IB through IB+NUM-1 in the file FNAM2 are moved to the file FNAM1. Cut operates on all columns of the file.

Example

```
>CUT SHORT < FILE 2 7

FILE
      11.0000     21.0000     31.0000
      12.0000     22.0000     32.0000
      13.0000     23.0000     33.0000
      14.0000     24.0000     34.0000
      15.0000     25.0000     35.0000
      16.0000     26.0000     36.0000
      17.0000     27.0000     37.0000
      18.0000     28.0000     38.0000
      19.0000     29.0000     39.0000
      20.0000     30.0000     40.0000

SHORT
      12.0000     22.0000     32.0000
      13.0000     23.0000     33.0000
      14.0000     24.0000     34.0000
      15.0000     25.0000     35.0000
      16.0000     26.0000     36.0000
      17.0000     27.0000     37.0000
```

INSI

Purpose

Generates signals in columns of a time series file. The signals may be of the following types:

```
PRBS  - a pseudorandom binary sequence
NORM  - a pseudo random signal with normal (gaussian)
        distribution
RECT  - a pseudo random signal with rectangular distribution
SINE  - a sine wave
ZERO  - a signal identically zero
STEP  - a step signal
RAMP  - a ramp signal
PULSE - a pulse signal
SRTW  - a sequential random telegraph wave
```

Command

```
INSI FNAME [(C)] NP [TSAMP]

        FNAME    - output file name
        C        - column number (default 1)
        NP       - number of data points wanted
        TSAMP    - sample interval in seconds
                   (default DELTA.)
```

Subcommands

```
PRBS [IBP [NBIT [ISTART [OPT] ] ] ]

        IBP      - basic period (1)
        NBIT     - number of bits in shift register
                   min 3, max 16 (7)
        ISTART   - specifies starting point in the sequence
                   1,2 or 3 (1)
        OPT      - trick option = 'KNEP'/'VOID' ('VOID')
                   KNEP  - F.O.A.-trick is used
                   VOID  - no trick
NORM [RMEAN SIGMA]
        RMEAN    - mean value (0.0)
        SIGMA    - std. dev. (1.0)
RECT [A B]
        A        - lower boundary (0.0)
        B        - upper boundary (1.0)
SINE [OMEGA FI]
        OMEGA    - frequency (1.0 radian/s)
        FI       - phase (0.0 degrees)
ZERO
STEP
RAMP [A B]
```

```
          A        - constant term (0.0)
          B        - linear term (1.0)
PULSE [LENGTH]
          LENGTH   - pulse length in samples (1)
SRTW [PS]
          PS       - change-of-sign probability (0.5)
LOOK      - display names of subcommands and reserved
            variables
KILL      - abort the operation of INSI and resume
            main command mode
X         - let the previous subcommands take effect,
            then resume main command mode
```

Function

Sequencies corresponding to the specifying subcommands are generated and upon the receipt of the command X they are stored in the specified file.

INSI makes implicit use of the reserved global variables IFP., NU., AMP., and DELTA.. IFP. specifies in all cases the sample point where the sequence should start, all previous values are set to zero. NU. is the state of the internal random number generator, and is automatically updated by INSI. It is sometimes desirable to save and/or initialize NU. using the LET command in order to obtain reproducible results. AMP. is used as the amplitude of some signals, see below. DELTA. is the value of the sampling interval to be recorded in the file.

Particulars:

PRBS
This signal is generated as the output of a shift register with feedback. The shift register is clocked at each IBP:th sampling point. NBIT specifies the length of the shift register. The output repeats itself after $2^{NBIT}-1$ clock instants. Four different initial values, i.e. the starting points in the periodic sequence, can be chosen by ISTART. The last argument, the "trick" parameter KNEP, specifies a slightly different form of the PRBS output; it is the output of a flip-flop complementing at the clock instants if the original sequence is negative. Thus the new sequence has asymptotically zero mean value, which the standard PRBS has not.

RECT
A rectangularly distributed pseudo random signal is generated by a mixed congruential method. NU. is used and altered.

NORM
An approximately normally distributed (i.e. gaussian) pseudorandom signal is generated as the sum of twelve rectangular values as above. NU. is used and altered.

SRTW
This is a two-valued (+a or -a) signal with the amplitude (a) specified by AMP. and with a given change-of-sign probability. Changes of sign occur when a rectangularly distributed (0,1) random number as above computed at each sample point falls below the parameter specified in the command. NU. is used and altered.

SINE
The phase refers to the phase at the starting point (i.e. IFP.). Thus FI = π/2 = 1.570796 will give a cosine signal. AMP. is the amplitude.

ZERO
The signal is constantly zero.

STEP
The signal changes from zero to AMP. at the sampling point IFP..

RAMP
Starting at sampling point IFP., a signal of the form B*(i-IFP)+A is generated, where i is the number of the sampling point.

PULSE
A pulse of amplitude AMP. starts at sampling point IFP. and lasts a specified number of sampling intervals.

Cautions: Restrictions

If the file specified in the command already exists, it will be changed or expanded, provided that the length and column number arguments are compatible with the old file.

Hints

a) The global variables IFP., NU. etc. are of course accessible by the commands LET, READ, and WRITE, also between subcommands.

b) More complicated signals can be constructed using commands like CUT, CONC, SCLOP, and VECOP on the results of INSI.

Example

The following operations yield the signal shown below (see Figure 1).

```
>INSI  T  30
    >LET IFP.=10
    >RAMP 0. 0.2
    >LET IFP.=20
    >RAMP -2. -0.2
    >X
>VECOP U < T(1) + T(2)
>SCLOP U < U + 0.5
>PLOT (30) U 0 2.5
```

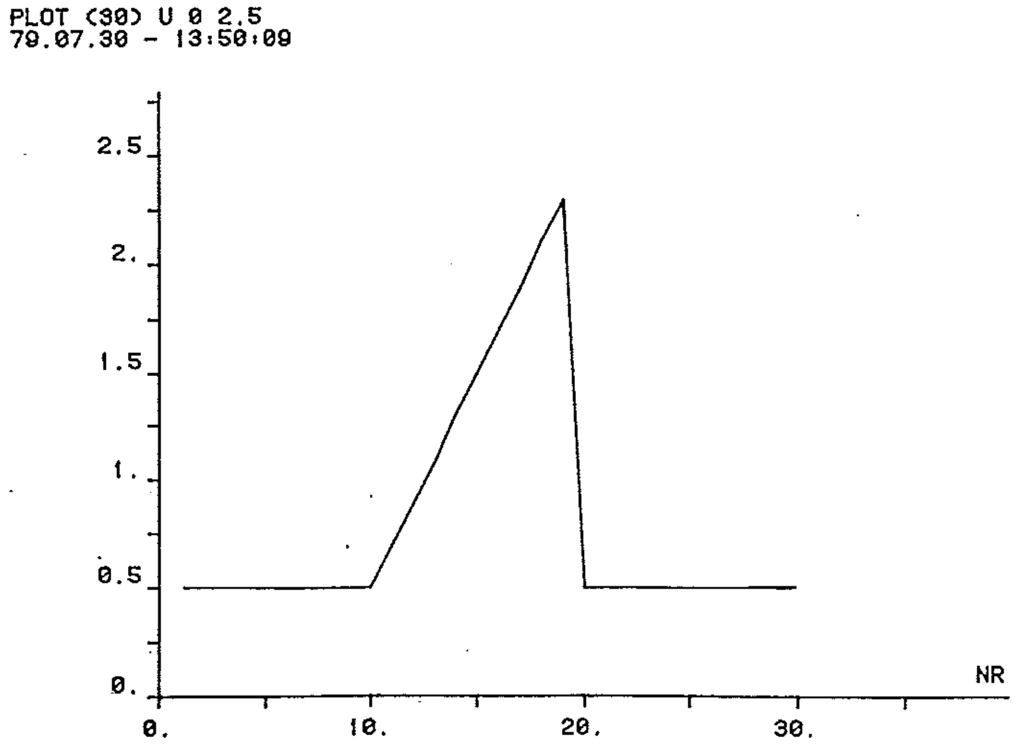


Figure_1

PICK

Purpose

To pick out equidistant samples from a time series file.

Command

```
PICK FNAM1 < FNAM2 N

FNAM1    - output file name
FNAM2    - input file name
N        - each N:th record in FNAM2 is written into FNAM1
```

Function

Each N:th sample in the file FNAM2 is transferred to the file FNAM1. PICK operates on all columns of the file.

Cautions: Restrictions

Be cautious not to violate the sampling theorem conditions in using PICK.

SCLOP

Purpose

To perform scalar operations on a data vector such as amplitude scaling or offset correction.

Command

```
SCLOP [FNAM1[(C1)] < FNAM2[(C2)] OPER CONST
```

FNAM1	- name of output file (default FNAM2)
C1	- column number for output file (default 1)
FNAM2	- name of input file
C2	- column number for FNAM2 (default 1)
OPER	- operation to be performed = '+', '-', '*' or '/'
CONST	- variable or unsigned numerical constant preceded by a space

Function

Each element in column C2 of FNAM2 is added, subtracted, multiplied or divided by CONST. The resulting data vector is placed in column C1 of FNAM1.

Hints

To subtract the mean value of a data vector use STAT to compute the mean and then SCLOP to subtract it. This can also be done as a 0:th order trend correction by TREND.

STAT

Purpose

To compute the statistical properties sum, mean, variance, standard deviation, minimum, and maximum for a time series.

Command

```
STAT FNAME [(C)] [EXT]

FNAME   - name of data file
C       - column number (default: 1)
EXT     - name extension for global variables
```

Function

The sum, mean, variance, standard deviation, minimum, and maximum for the C:th (default 1st) column in the file FNAME are computed and displayed. The results will also be printed on line printer if the reserved variable PRINT. is nonzero. The output on the terminal will appear only if the switch 'TEXT' is 'ON'.

Those of the global variables SUM.EXT, MEAN.EXT, VAR.EXT, STDEV.EXT, MIN.EXT, and MAX.EXT that are previously defined as real variables will receive the appropriate value, provided that EXT is specified in the command.

Example

```
>STAT UT(2)

    UT(2)
    SUM      = -216.508
    MEAN     = -2.16508
    VARIANCE =  424.615
    ST.DEV.  =  20.6062
    MINIMUM  = -43.0363
    MAXIMUM  =  48.3345
    LENGTH   =  100
```

VECOP

Purpose

To add, subtract, multiply, or divide two data vectors element by element.

Command

```
VECOP [FNAM1[(C1)] < FNAM2[(C2)] OPER FNAM3[(C3)]

FNAM1    - output file name (default FNAM2)
C1       - column number within FNAM1 (default 1)
FNAM2,3  - input file names
C2,3     - column numbers within FNAM2,3 (default 1)
OPER     - type of operation to perform = '+', '-', '*',
           or '/'
```

Function

Each element in column C2 of FNAM2 is added, subtracted, multiplied, or divided by the corresponding element in column C3 of FNAM3. The resulting data vector is placed in column C1 of FNAM1.

ALTER

Purpose

To alter the elements of a matrix.

Command

```
ALTER [AGGREG:] MATRIX [(IR IC) VALUE]

AGGREG  - aggregeate file name
MATRIX  - matrix file name
IR      - row index
IC      - column index
VALUE   - new value
```

Subcommands

```
KILL     - resumes main command mode, MATRIX is not updated
X        - resumes main command mode, MATRIX is updated
IR IC VALUE - change element (IR,IC) to VALUE
```

Function

This command has two formats. If the matrix is specified together with row and column index and the new value in the same line, the entire operation is performed in one step, i.e. the matrix is read and written back to the data base. If only the matrix is specified, row and column indices and new values are expected as subcommands. Not until the execute command (X) is received, the matrix is written to the data base.

Cautions, Restrictions

If more than one matrix element is to be altered, the subcommand form is much more efficient.

EIGEN

Purpose

To compute (and display) the eigenvalues and eigenvectors of a matrix.

Commands

```
EIGEN [/['EVAL'] ['EVEC']/] [LF] [MF] < [AG:]MAT

LF      - locus file receiving the eigenvalues('EVAL')
MF      - matrix file receiving the eigenvectors ('EVEC')
AG      - aggregate file name
MAT     - matrix file name
```

Function

The eigenvalues and eigenvectors of the specified matrix is computed, and if the switch GRAPH is ON, the eigenvalue positions are indicated on the terminal display. If the switch TEXT is ON, they are also given in numeric form.

The eigenvalues and/or eigenvectors may also be output as a locus file resp. a matrix file. The file indication flags EVAL resp. EVEC are used to indicate the type of output desired.

Method

The transformation matrix is determined in three steps. First a transformation is determined that balances the given A-matrix. This is done to improve numeric preccision since it is known that errors in the eigenvalues usually are proportional to the norm of the matrix. The second step consists of a transformation to upper Hessenberg form. Finally, the matrix is transformed using the QR method to triangular form where eigenvalues and eigenvectors are readily available.

To avoid complex numbers, the triangulation is allowed to leave 2x2 blocks on the diagonal representing a pair of complex conjugated eigenvalues.

The eigenvectors are output in the columns of the matrix specified by EVEC, while the eigenvalues are in the locus file specified by EVAL. To a real eigenvalue corresponds a real eigenvector. For a complex eigenvalue, the real part of the eigenvector is stored in the first corresponding column while the imaginary part of that eigenvector is stored in the second corresponding column. The second eigenvector corresponding to that complex conjugated pair is not output but is known to be the complex conjugate of the first one. The representation of eigenvectors used has the advantage that if this eigenvector matrix is used as T^{-1} in the similarity transformation $D = TAT^{-1}$, the matrix D is diagonal except for complex conjugate eigenvalue pairs x ± iy which are represented as diagonal 2x2 blocks of the form:

$$\begin{bmatrix} x & y \\ -y & x \end{bmatrix}$$

Thus D is a real matrix. Cf. the example below.

References

G. Peters, J.H. Wilkinson: Eigenvectors of real and complex matrices by LR and QR triangularisations. In J.H. Wilkinson, C. Reinsch: Linear Algebra, Springer-Verlag, 1971.

Hints

a) The eigenvalues may also be plotted by the command PLEV which also allows changes to be made.

b) The eigenvalues (poles) of a system may be computed through the command POLES.

Examples

The following matrix A is given:

$$\begin{bmatrix} -3 & 0 & 0 & 0 \\ 1 & -2 & -2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -0.5 \end{bmatrix}$$

The commmand

```
    >EIGEN  <  A
```

will then give the plot shown in Figure 1, plus a table with the respective numeric values on the terminal display.

The command

```
>TURN GRAPH OFF
>TURN TEXT OFF
>EIGEN /EVEC/ M < A
```

will produce no terminal output but the matrix M will look like

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & -0.6 \\ 0 & -0.5 & -0.5 & 0.2 \\ 1 & -0.2 & 0.6 & -0.08 \end{bmatrix}$$

Note that the order of the eigenvector and eigenvalue output files is determined through the file identification flags:

```
>EIGEN /EVEC EVAL/ M LF < A
```

gives the eigenvectors in the matrix M and the eigenvalues in the locus file LF.

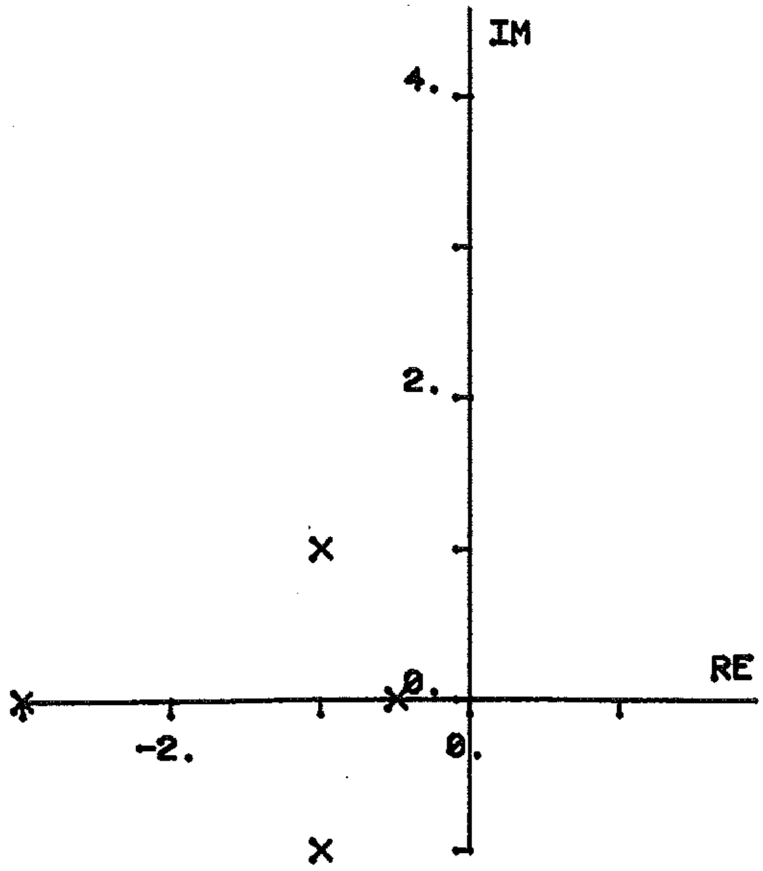


Figure 1

ENTER

Purpose

To enter a matrix, element by element.

Command

```
ENTER [AG:]MAT NR [NC] [TSAMP]
```

AG	- name of aggregate file
MAT	- name of matrix file
NR	- number of rows
NC	- number of columns (default NR)
TSAMP	- sample interval (default DELTA.)

Subcommands

KILL	- resume main command mode, no action taken
X	- resume main command mode, MAT is updated/generated

Function

After the command line has been entered, the values of the matrix elements are asked for. They may be entered in free format on one or several lines. An error message is generated if too many values are given. The subcommand KILL may be given if the user wants to abort the operation. It may be used in place of any element value, but it must appear at the beginning of a line. The subcommand X is legal only when all element values have been entered.

Cautions, Restrictions

Do not forget the specification of a sampling interval for a matrix to be part of a discrete time state space description. Cf. DELTA.

Examples

To enter a square (2x2) matrix with sample interval 3 s.

```
>ENTER  A  2  3.
ROW 1
    #> 1 2
ROW 2
    #> 3 4
    #> X
>LIST  A
```

Matrix Operations
ENTER

```
    LIST  A

    1.000     2.000
    3.000     4.000
```

To enter a 2x3 matrix and changing one's mind after the 2 2 element:

```
>ENTER  B  2  3
ROW 1
    #> 1 2
    #> 3
ROW 2
    #> 4 5
    #> KILL
>
```

EXPAN

Purpose

To generate a matrix made up of blocks of other matrices.

Command

```
EXPAN [[AG1:]M1] < [AG2:]M2[(IX2 IY2)] [[AG3:]M3[(..)]..]

AG      - aggregate file name
M       - matrix file name
IXi,IYi - the coordinates in the new matrix for
          the upper left corner of matrix Mi
          (by default IX=IY=1)
```

Function

A new matrix is generated from one or more old matrices. Undefined elements are zeroed. The position of the old matrices within the new one is specified by means of the location of its upper left corner (the 1 1 element) in the new matrix. The dimension of the new matrix is determined by the maximum row and column position occupied by any lower right corner of the old matrices.

Example

Let M1 be a 3x3 matrix of threes and M2 a 2x2 matrix of twos. Then

```
>EXPAN  X1 < M2 (1  2)
```

yields

```
     X1:  0  2  2
          0  2  2
```

```
>EXPAN  X2 < M1 (1  1)  M2 (2 1)
```

yields

```
     X2:  3  3  3
          2  2  3
          2  2  3
```

MATOP

Purpose

To evaluate matrix expressions.

Command

```
MATOP [(EXT)] [[AGGREG:]MATRIX] < algebraic matrix expression

EXT     - name extension for global variables
AGGREG  - aggregate file name
MATRIX  - name of resulting matrix file (by default MATRIX =
          the 1st matrix name in the algebraic expression)
```

Function

The right hand side matrix expression is evaluated according to the rules stated below. The value is assigned to the matrix given in the left hand side (possibly a component of an existing aggregate), or if no left hand side was given, to the first matrix specified in the right hand side.

If EXT was included in the command line, those of the global variables DET.EXT, MINMAX.EXT, and TRACE.EXT that exist as real variables will receive the determinant, the minmax norm, and the trace of the result, respectively.

Method

The right hand side expression is evaluated from left to right following standard precedence rules. Parentheses are allowed. Below a formal definition of the allowed expression follows. Some very simple production rules are omitted.

```
<adding operator>::= +/-

<multiplying operator>::= *

<function operator>::= TR/PSINV/^<signed integer>

<matrix reference>::= <aggregate name>:<matrix name>/
                      <matrix name>

<scalar>::= <variable reference>/<real constant>

<primary>::= <matrix reference>/<scalar>
             <multiplying operator><matrix reference>

<factor>::= <primary>/<primary><function operator>/
            (<expression>)
```

```
<term>::=<factor>/<term><multiplying operator><factor>

<expression>::= <term>/<adding operator><term>/
                <expression><adding operator><term>
```

Examples

Some simple expressions follow.

```
2*A
A+B
A*B+C
A^-1
0.5*(A TR + A)
```

REDUC

Purpose

To pick out a block from an existing matrix.

Command

```
REDUC [[AG1:]M1] < [AG2:]M2 (IX1 IY1 IX2 IY2)
```

```
AG      - aggregate file name
M       - matrix file name
IX1,IY1 - indices for the upper left corner of
          the part to be saved
IX2,IY2 - indices for the lower right corner of
          the part to be saved
```

Function

A new matrix is generated as a block of an old one. The block is specified in terms of its upper left and lower right corner.

Example

The matrix X2 (cf. EXPAN) is given

```
     X2:  3  3  3
          2  2  3
          2  2  3
```

```
>REDUC  X3 < X2 (2  2  3  3)
```

yields

```
     X3:  2  3
          2  3
```

UNITM

Purpose

To generate a unit matrix.

Commands

```
UNITM [* FACTOR] [AG:]MAT NR [TSAMP]
```

FACTOR	- scale factor (by default FACTOR = 1)
AG	- name of aggregate file
MAT	- name of matrix file
NR	- number of rows
NC	- number of columns (default NR)
TSAMP	- sample interval (by default TSAMP = DELTA.)

Function

A unit matrix, optionally scaled according to the scale factor given, is generated.

Examples

```
>UNITM  A  2
```

results in

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
>UNITM  *  -1  A  3
```

results in

$$A = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

ZEROM

Purpose

To generate a zero matrix.

Commands

```
ZEROM [+ TERM] [AG:]MAT NR [NC] [TSAMP]

TERM    - constant term (by default TERM = 0)
AG      - name of aggregate file
MAT     - name of matrix file
NR      - number of rows
NC      - number of columns (default NR)
TSAMP   - sample interval (by default TSAMP = DELTA.)
```

Function

A zero matrix of the specified dimension is generated. Optionally, a specified constant may be added to all elements.

Examples

```
>ZEROM   B  2
```

results in

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

```
>ZEROM  +  -1  B  1  4
```

results in

$$B = [-1 \quad -1 \quad -1 \quad -1]$$

CONT

Purpose

To transform a system to Continuous State Space Form from Discrete State Space Form.

Command

```
CONT [SYSOUT][(NAMOUT)] < SYSIN[(NAMIN)] [EPS]

SYSOUT   - name of system file for output system
           by default SYSOUT = SYSIN
NAMOUT   - name of section within SYSOUT
SYSIN    - name of system file for input system
NAMIN    - name of section within SYSIN
EPS      - test quantity
           by default EPS = the reserved variable REPS.
```

Function

The matrices of the input system is read and transformed to their continuous time values. Any structure given, e.g. division of inputs into control inputs and disturbance inputs, is maintained.

Method

We have the following notations in the continuous and discrete time cases:

$$\dot{x} = A_C x + B_C u \qquad x_{t+T} = A_D x_t + B_D u_t$$

$$y = C_C x \qquad y_t = C_D x_t$$

where A_D and B_D are computed from A_C and B_C as (cf. the command SAMP):

$$A_D = e^{A_C T} \tag{1}$$

$$B_D = \int_0^T e^{A_C s} ds\, B_C \tag{2}$$

Conversely we have:

$$A_C = \frac{1}{T} \ln A_D \qquad (3)$$

$$B_C = (A_D - I)^{-1} A_C B_D. \qquad (4)$$

The logarithm in (3) is computed as:

$$Y = A_D^{1/2^n}$$

where the integer n is chosen so that $\|Y-I\| < 1$.

$$X = A_D^{1/2}$$

is found by solving $X^2-A_D = 0$ by an iterative technique. Then by solving $e^Z-Y = 0$ also by an iterative technique A_C is found:

$$Z = \ln Y$$

$$A_C = \frac{2^n}{T} \ln A_D^{1/2^n} = \frac{1}{T}\cdot 2^n \cdot Z = \frac{1}{T} \ln A_D.$$

In evaluating (4) directly, a problem would be encountered for systems containing integrators since A_D-I would be singular. Therefore (4) is evaluated as follows:

$$B_C = (A_D-I)^{-1} A_C B_D = (e^{A_C T}-I)^{-1} A_C B_D =$$

$$= (A_C T\, \Psi(A_C T))^{-1} A_C B_D = \frac{1}{T} \Psi^{-1}(A_C T)\, B_D,$$

where $\Psi(A_C T)$ is defined by the series expansion

$$\Psi(A_C T) = I + \frac{1}{2!} A_C T + \frac{1}{3!} A_C^2 T^2 + \frac{1}{4!} A_C^3 T^3 + \ldots$$

POLES

Purpose

To compute (and display) the poles and eigenvectors of a state space system.

Commands

```
POLES [/['POLE'] ['EVEC']/] [LF] [MF] < SYST[(NAME)]
```

LF - locus file receiving the poles ('POLE')
MF - matrix file receiving the eigenvectors ('EVEC')
SYST - system file name
NAME - name of section within SYST

Function

The matrix A of a state space system representation is read and its eigenvalues and eigenvectors are computed.

For further comments, refer to the command EIGEN.

SAMP

Purpose

To compute the discrete time (i.e. sampled) form of a continuous time state space system description.

Command

```
SAMP [SYSOUT][(NAMOUT)] < SYSIN[(NAMIN)]

SYSOUT  - name of system file for output system
          by default SYSOUT = SYSIN
NAMOUT  - name of section within SYSOUT
SYSIN   - name of system file for input system
NAMIN   - name of section within SYSIN
```

Function

The matrices of the input system description are read. Any structural information, i.e. presence of input/output matrix blocks, is noted and used in the output of the result.

The system matrices are then transformed to a discrete time description according to the sample interval found in the output system description.

Method

The input system matrices are blocked together to a system description of the form:

$$\begin{cases} \dot{x} = A_C x + B_C u \\ \\ y = C_C x + D_C u \end{cases}$$

These equations are then transformed into

$$x_{t+T} = A_D x_t + B_D u_t$$

$$y_t = C_D X_t + D_D u_t$$

where

$$A_D = e^{A_C T} = A_C T \, \Psi(A_C T) + I$$

$$B_D = \int_0^T e^{A_C s} ds\ B_C = \Psi(A_C T)\ B_C\ T$$

$$\Psi(A_C T) = I + \frac{1}{2!} A_C T + \frac{1}{3!} A_C^2 T^2 + \ldots$$

$$C_D = C_c$$

$$D_D = D_c$$

(cf. the command CONT). $\Psi(A_c T)$ is computed from its series expansion.

<u>Reference</u>

C. Kallstrom: Evaluation of e^A and $\int e^{As} ds$. Report TFRT-3053, Dept. of Automatic Control, Lund Institute of Technology, Lund, Sweden.

SIMU

Purpose

To simulate a dynamic system in state space representation with specified inputs. A continuous time system representation is automatically converted to its discrete time version for the purpose of the simulation.

Command

```
SIMU [/['Y'] ['Z'] ['X']/] [Y] [Z] [X] <
     SYST[(NAME)] [/['U'] ['W'] ['V'] ['E']/] [U] [W] [V] [E]
```

Y	- measured outputs
Z	- control outputs
X	- states
SYST	- name of system file
NAME	- name of section within SYST
U	- control inputs
W	- disturbance inputs
V	- stochastic inputs
E	- measurement error signals

Note: e.g. Y, Z, U etc. may include a column specification, i.e. Y(1), Z(Cz).

Function

The system description matrices are read and, if on continuous time form, converted to discrete time form with the sample interval of the inputs. A set of input time series files corresponding to the inputs used in the system must be given as input arguments. They should be given in order of appearance in the system description, otherwise file identification flags must be used to identify the inputs.

The output of SIMU is up to three time series files selected through the file identification flags. They will contain the measured output Y, the controlled output Z or the state K.

All signal specification arguments may contain a column specification, e.g. Y(1), specifying the first of the respective signal.

If no initial state is specified, it is assumed zero.

Method

A continuous time system representation is converted to discrete time form using the same algorithm that is used in the command SAMP. Once the discrete time state space equations are given, they are used without further transformations to successively compute states and outputs.

Examples

If a system S was given by the state equations

$$\begin{cases} \dot{x} = Ax + Bu + B_v v \\ y = Cx + B_e e \end{cases}$$

a command simulating this system could be

```
     >SIMU /Y X/ F1 F2 < S  U  V  E
```

The result is thus two data files F1 and F2, where F1 contains the measured outputs and F2 the state, as indicated by the output file identification flags.

If the user of some reason would like to specify the inputs in an order different from the one used in the system description S, input file identification flags would have to be used, as in:

```
     >SIMU  Y < S  /V E U/  NOISE(1)  NOISE(4) U
```

SIMU is used in the examples found for OPTFB and REDFB.

SPSS

Purpose

To compute the frequency characteristics (power spectrum or amplitude and phase) between one input and one output for a system on state space form.

Command

```
SPSS [('POW'/'AMP')] FRF[(F)] < SYST[(NAME)] NY NU [FREQ]
```

'POW'/'AMP'	- switch choosing a power spectrum or an amplitude and phase computation default is 'AMP'
FRF	- frequency response file
F	- frequency response number (default value 1)
SYST	- system file name
NAME	- section name of system file
NY	- output (measurement) signal number
NU	- input (control) signal number
FREQ	- file with frequency values in the first column

Function

The frequency response between input NU and output NY is computed for the system description specified. The system description may be on continuous time or discrete time form.

The frequency points are logarithmically distributed between the two values WMIN. and WMAX. (two reserved global variables), or if the file FREQ is present, the frequency points are taken from its first column.

Method

In the continuous time case, the evaluation of the frequency response amounts to computing the matrix $H(j\omega)$ for different values of ω.

$$H(j\omega) = C(j\omega-A)^{-1} B + D.$$

To avoid computation with complex matrices we reformulate the expression to

$$H(j\omega) = - C(\omega^2 I + A^2)^{-1} (j\omega I + A) B + D.$$

Here, the real and imaginary parts may be computed separately.

In the discrete time case, the formulas look like:

$$H(z) = C(zI-A)^{-1}B + D; \quad z = e^{j\omega T} = \cos \omega T + j \sin \omega T.$$

The real and imaginary parts are:

$$\mathrm{Re}\, H(e^{j\omega T}) = C(A^{2}+ I - 2 \cos(\omega T)A)^{-1}(I \cos(\omega T)-A) B + D$$

$$\mathrm{Im}\, H(e^{j\omega T}) = - C(A^{2}+ I - 2 \cos(\omega T)A)^{-1} \sin(\omega T) B.$$

Knowing the real and imaginary parts, it is a simple matter to compute the power spectrum or the amplitude and phase information. The phase is expressed in degrees and an effort is made to make it continuous across the ± 180° boundaries.

SYSOP

Purpose

To generate a system formed from a number of subsystems.

Command

```
SYSOP ST[(NAMT)] < [/F1/]SU1[(NSU1)] [[/F2/]SU2[(NSU2)] .. ]

ST       - name of system file for the total system
NAMT     - name of section within ST
Fi       - i:th subsystem flag = 'S'/'M' (default: 'S')
           S      - the i:th subsystem should be a complete
                    dynamical system
           M      - the i:th subsystem should only consist of
                    a matrix, e.g. a feedback gain matrix
SUi      - Fi = 'S': mame of system file for the i:th
                     subsystem
           Fi = 'M': name of matrix
NSUi     - name of section within SUi, valid only if Fi = 'S'
```

Subcommands

```
LOOK     - displays the dimensions of the systems
KILL     - aborts SYSOP
X        - effectuates SYSOP
IN 'U'/'W'/'V'/'E'i[(Ci1 Ci2 .. )] <
    expression (itself, total inputs, sub-states, other
    sub-outputs) specifies SUi's input(s)
OUT 'Y'/'Z'[(Ci1 Ci2 .. )] <
    expression(itself,inputs,states,sub-outputs)
    specifies ST's output(s)

    expression is a linear combination of inputs, states and
    outputs with terms on the following form:
    .. '+'/'-'[WEIGHT*]SIGMNEM-concatenated-with-i
                                             [(Ci1 Ci2 .. )] ..

WEIGHT   - weighting factor (default: 1)
SIGMNEM  - signal mnemonic = 'U'/'W'/'V'/'E' for inputs
           U      - control inputs
           W      - disturbance inputs
           V      - stochastic inputs
           E      - measurements error signals

         - signal mnemonic = 'X' for states

         - signal mnemonic = 'Y'/'Z' for outputs
           Y      - measured outputs
           Z      - controlled outputs
```

i — subsystem index, i.e. simply the position in the right part of the main command where a subsystem was referenced

Ci1 .. — optional column numbers, useful if not all the components of a signal vector are wanted or if a permutation of them is desired (default: all the components unpermuted)

Function

The right hand side list of subsystems is read and decoded. The subsystems may be either state space representations or matrices (e.g. feedback or filter gain matrices). At least one subsystem must contain a state equation. All subsystems must be of the same type, discrete time or continuous time.

The way the subsystems are connected is specified through subcommands. The dimensions of the available signals for the different subsystems may be viewed through the subcommand LOOK.

- To refer to a subsystem, the signal mnemonic should be concatenated with that subsystem's index, e.g. the 3rd subsystem's control inputs are referred to as U3.

- Expression may also consist of only the number zero, in which case all the signals on the left side will be identically zero.

- The 'IN'/'OUT' commands may be accumulative, i.e. a previously defined signal may be superimposed onto itself. In this case the input may only be specified once, i.e. as the 1st term on the right side of the command string. This is practical for two reasons:

  1) 'IN'/'OUT' may be continued over many lines,

  2) an erroneous 'IN'/'OUT' contribution may be annihilated without the aid of 'KILL'.

- The dimensions of the left and right hand expressions must agree.

- At least one 'OUT' command must be issued before 'X'.

- Resulting algebraic loops will cause a warning message.

- ST's inputs must be referenced at least once by an 'IN' command.

- At least one subsystem must be a dynamical system.

- The output signals from a non-dynamical subsystem are treated as measured outputs, 'Y'.

- The reserved variable AEPS. is used as an absolute test quantity to detect algebraic loops.

- The reserved variable REPS. is used as a relative test quantity to decide whether the compound system is realizable or not.

- Zero result matrices will not be output.

Method

The matrices of the new total system are formed as block-diagonal matrices from the corresponding matrices in the subsystems. (In the following discussion, continuous time equations are assumed and blocking of inputs and outputs into controlled ... measured ... etc. is neglected.) Thus we have with x, u, and y the concatenation of state, input and output vectors:

$$\begin{cases} \dot{x} = \tilde{A}x + \tilde{B}u \\ y = \tilde{C}x + \tilde{D}u \end{cases} \qquad (1)$$

where $\tilde{A}$ for instance is given as

$$\tilde{A} = \begin{bmatrix} A_1 & & \\ & A_2 & \\ & & \ddots \end{bmatrix}$$

The A_i being the A matrices of the subsystem 1, 2,... .

The connecting information from the subcommands are condensed into the connect matrices K_i:

$$\begin{cases} y = K_1 y + K_2 x + K_3 u_T \\ y_T = K_4 y + K_5 x + K_6 u_r \end{cases} \qquad (2)$$

The index T denotes the input resp. output of the total system to be formed.

Solving for u in (1) and (2) and inserting into (1) gives the matrices of the total system:

$$A_T = \tilde{A} + \tilde{B} (I - K_1\tilde{D})^{-1} (K_1\tilde{C} + K_2)$$

$$B_T = \tilde{B} (I - K_1\tilde{D})^{-1}$$

$$C_T = K_4 [\tilde{C} + \tilde{D} (I - K_1\tilde{D})^{-1} (K_1\tilde{C} + K_2) + K_5]$$

$$D_T = K_4\tilde{D} (I - K_1\tilde{D})^{-1} K_3 + K_6$$

The matrix $I - K_1\tilde{D}$ is tested for algebraic loops. It should be triangularizable using permutations only, otherwise a warning message is given. If it is non-invertible, SYSOP naturally fails.

Cautions, Restrictions

Note that SYSOP generates a system file with name ST[(NAMT)]. This file must not exist previously! if it does, delete it with DELETE.

Hints

a) Note that the order of state variables in the total system is the order in which subsystems were specified.

b) In many cases, the connecting of subsystems is an operation performed unchanged many times in the solution of a problem. SYSOP with its subcommands is thus very naturally performed in a macro.

c) SYSOP is very frequently used. E.g. it is used to form the closed loop system after a call to OPTFB and a Kalman filter after a call to KALFI.

Examples

The following example is intended to give an illustration of some of the possible subcommand forms available in SYSOP. It is by no means typical for the normal use.

For some other uses of SYSOP, refer to the examples used in OPTFB, PENLT, and KALFI.

Assume that the systems S1, S2, and S3 are given and that L is a (feedback) matrix. Figure 1 illustrates the output from the LOOK subcommand. The following command sequence then connects the subsystems.

```
>SYSOP  ST < S1  S2  S3  /M/  L
   >LOOK
   >IN  U1 < U
   >IN  U2 < Y1 + Y4
   >IN  W2 < Y3
   >IN  U3 < Y1 + Y4
   >IN  V3 < V
   >IN  E3 < E
   >IN  U4(1-3) < X1(1-3) - X2(1-3)
   >IN  U4(4 5) < - X2(4 5)
   >OUT  Y < Y3
   >OUT  Z < 0.1 * Y1 + 0.1 * Y4
   >LOOK
   >X
>
```

The result of the second LOOK subcommand is shown in Figure 2. Note that the dimensions of the total system now is complete.

```
SYSOP ST<S1 S2 S3 /M/ L

                          ST          S1          S2          S3
                         (       )   (       )   (       )   (       )
CONTROL      INPUTS                   U1    2     U2    2     U3    2
DISTURBANCE  INPUTS                               W2    3
STOCHASTIC   INPUTS                                           V3    5
MEASUREMENT  ERRORS                                           E3    3
             STATES       X    13     X1    3     X2    5     X3    5
MEASURED     OUTPUTS                  Y1    2     Y2    5     Y3    3

                          L

CONTROL      INPUTS       U4   5
MEASURED     OUTPUTS      Y4   2
```

Figure__1. The output from the command LOOK before any connecting commands have been issued. (The empty parentheses would have contained section names, if any were given.)

```
SYSOP ST<S1 S2 S3 /M/ L

                          ST
                         (       )   (       )   (       )   (       )
CONTROL      INPUTS       U    2      U1    2     U2    2     U3    2
DISTURBANCE  INPUTS                               W2    3
STOCHASTIC   INPUTS       V    5                              V3    5
MEASUREMENT  ERRORS       E    3                              E3    3
             STATES       X    13     X1    3     X2    5     X3    5
MEASURED     OUTPUTS      Y    2      Y1    2     Y2    5     Y3    3
CONTROLLED   OUTPUTS      Z    2

                          L

CONTROL      INPUTS       U4   5
MEASURED     OUTPUTS      Y4   2
```

Figure__2. The output from the command LOOK after all connecting commands have been received.

SYST

Purpose

To aid in the generation of system description files.

Command

```
SYST [(SUBSW)] SYSNAM[(SECNAM)] [< [(SYSTYP)] [SYSMNEM]
     [DT] [AGRNAM] [(TIMTYP)/OP/LAMVAL] [ATRNAM]]

SUBSW   - subcommand switch = 'SC'/'VOID' (default: 'VOID')
          SC     - subcommands wanted
          VOID   - no subcommands wanted
SYSNAM  - name of resulting system file
SECNAM  - see subcommand BEGIN
SYSTYP  - system type = 'SS'/'MTF'/'PM' (default: 'SS')
          SS     - State Space
          MTF    - Miso Transfer Function
          PM     - Polynomial Matrix
SYSMNEM - system mnemonic, a short form used to specify
          the system equation
          SS : SYSMNEM = 'ABC'/'ABCD'/'ABCXO'/'ABCDXO',
               where A, B, C etc. denote system matrices
               (default: 'ABC')
          MTF: SYSMNEM = 'AB'/'ABC'/'ABCI' (default: 'AB')
          PM : SYSMNEM = 'TUUV'/'TUUVWU' (default: 'TUUV')
DT      - see subcommand TSAMP (default DELTA.)
AGRNAM  - see subcommand AG
TIMTYP  - see subcommand TIME
OP      - see subcommand SHIFT
LAMVAL  - see subcommand LAMBDA
ATRNAM  - see subcommand AT
```

Subcommands

```
BEGIN [SECNAM]  defines a section name
   SECNAM       - section name (default: missing)

TSAMP DT   defines a sample interval
   DT      - sample interval expressed in seconds

LOOK       displays the current contents of SYSNAM[(SECNAm)]

AG [(AGTYP)] [AGRNAM]   declares an aggregate file
   AGTYP   - aggregate type = 'S'/'L'/'C'/'E' (default: 'S')
             S  - system equation, the only valid one for
                  SYSTYPs differing from 'SS'
             L  - loss function
             C  - covariance function
             E  - extended loss function
   AGRNAM  - aggregate file name (default: SECNAM (SYSNAM if
                SECNAM is omitted) for the main command and
                AGTYP = 'S', otherwise missing)

AT [ATRNAM]     declares an attribute file name, valid
                only if SYSTYP = 'PM'
   ATRNAM       - attribute file name (default: missing)

TIME TIMTYP     defines whether a system is time variable or
                not, valid only if SYSTYP = 'SS'
   TIMTYP       - time switch = 'VAR'/'INV' (default: 'INV')
                  VAR  - time variable
                  INV  - time invariant

LAMBDA LAMVAL     defines lambda, i.e. the noise standard
                  deviation,valid only if SYSTYP = 'MTF' and
                  if SYSMNEM contains the letter 'C'
    LAMVAL        - the value of lambda (default: 1, if SYSMNEM
                    contains the letter 'C', else missing)

SHIFT OP      defines the shift operator,
              valid only if SYSTYP = 'PM'
    OP        - operator = '+'/'-' (default: '+')
                +   - forward shift
                -   - backward shift

INS MNEM [< NAME] inserts terms into the system equations
    MNEM           - matrix or polynomial mnemonic, see below
    NAME           - name of the matrix or polynomial
                     corresponding to MNEM (default: MNEM)

DEL MNEM      removes terms from the system equations
    MNEM      - matrix or polynomial mnemonic, see below

KILL    skips the previous subcommand sequence including
        the last SYST command, i.e. inhibits the generation
        of SYSNAM[(SECNAM)]

X       closes SYSNAM[(SECNAM)] with the current contents,
        then exits from SYST
```

Function

The command generates a new system description file or, if the specified output file already exists, a new section within it.

The command uses a series of defaults such that many of the desires encountered in normal use will be possible to satisfy with a single main level command. If more freedom is required, a series of subcommands is available to specify these details.

The subcommands serve to control the inclusion/deletion of various items in the system description and to allow the user to select names for aggregates and polynomials or matrices. Deletion is by specifying a blank name for e.g. an aggregate, or through the DEL command.

Note that the subcommand LOOK will at any time display the current form of the system description.

The different matrices and polynomials are referenced through mnemonics shown below. The mnemonics are also the default names of the respective items.

The main command argument SYSMNEM serves to choose between standard sets of these matrices/polynomials. These standard sets are also given below.

Matrix mnemonics (SYSTYP = 'SS'):

```
A       B       BW      BV      "DX/DT = .., or XNEW = ..
C       D       DW      DE      "    Y = ..
G       H       HW              "    Z = ..
XO                              "initial state vector

QO      Q1      Q12     Q2      "loss function

RO      R1      R12     R2      "covariance function

EQO     EQ1     EQ12    EQ2     "extended loss function
EQ3     EQ4     EQ5
```

For the different possibilities of SYSMNEM, the following is included:

ABC	A	B	C		
ABCD	A	B	C	D	
ABCXO	A	B	C	XO	
ABCDXO	A	B	C	D	XO

Polynomial mnemonics (SYSTYP = 'MTF'):

```
A     B     C     D     "Ay = Bu + Ce
I                       "initial output values
```

For the different possibilities of SYSMNEM, the following is included:

AB	A	B		
ABC	A	B	C	
ABCI	A	B	C	I

Polynomial_mnemonics_(SYSTYP)_=_'PM'):

```
T        UU       UW       UV       "..*X = ..
V        WU       WW       WE       "    Y = ..
G        HU       HW                "    Z = ..
```

For the different possibilities of SYSMNEM, the following is included:

```
TUUV        T     UU    V
TUUVWV      T     UU    V     WV
```

Examples

The following serves to illustrate the operation of SYST. The global variable DELTA. = 1.0 during these examples.

a) The two commands

```
>SYST S(CSS) < 0.
>LIST (T) S
```

produces:

```
LIST (T) S
79.10.05 - 16:46:08

BEGIN CSS

"SYST S(CSS)<0.
"79.10.05 - 16:45:53
"
CONTINUOUS STATE SPACE REPRESENTATION

DYNAMICS, AGGREGATE: CSS,

DX/DT = A*X + B*U

Y = C*X

END
```

while

b)
```
>SYST S2(DSS) < ABCD S2DSS
>LIST (T) S2
```

produces

```
        LIST(T)S2
        79.10.05 - 16:47:16

        BEGIN DSS

        "SYST S2(DSS)<ABCD S2DSS
        "79.10.05 - 16:47:06
        "
        DISCRETE STATE SPACE REPRESENTATION

        SAMPLE INTERVAL 1. S

        DYNAMICS, AGGREGATE: S2DSS,

        XNEW = A*X + B*U

        Y = C*X + D*U

        END
```

Note in case b) that the default value of the sample interval takes effect producing a discrete time state space representation, and that in both a) and b) the state space form is assumed. In b) a fourth system matrix (D) is included and the dynamics aggregate name is changed from its default, viz. the section name.

c) Here we use the subcommand switch:

```
   >SYST (SC) S3 (S3CSS)
       >LOOK
```

The subcommand LOOK gives the present form of the system description.

```
        "SYST(SC)S3(S3CSS)
        "79.10.05 - 16:48:46

        BEGIN S3CSS

        DISCRETE STATE SPACE REPRESENTATION

        SAMPLE INTERVAL 1. S

        DYNAMICS, AGGREGATE: S3CSS,

        XNEW = A*X + B*U

        Y = C*X

        END
```

d) Assuming we did not want a discrete time representation and intending to use a linear quadratic loss function, we do:

```
>TSAMP 0.
>AG (L) CLS3
>INS Q1
>INS Q2
>LOOK
```

We have now obtained the following:

```
"SYST(SC)S3(S3CSS)
"79.10.05 - 16:50:21

BEGIN S3CSS

CONTINUOUS STATE SPACE REPRESENTATION

DYNAMICS, AGGREGATE: S3CSS,

DX/DT = A*X + B*U

Y = C*X

LOSS FUNCTION, AGGREGATE: CLS3,

Q1: Q1, Q2: Q2

END
```

If this is what we wanted, we finish the subcommand sequence by:

```
>X
```

TRANS

Purpose

To transform a continuous time loss function or covariance function into discrete time form.

Command

```
TRANS [(SW)] SYSOUT[(NAMOUT)] < SYSIN[(NAMIN)]

SW       - problem switch = 'Q'/'R' (default: 'Q')
           Q: transform the loss matrices
              (control problem)
           R: transform the covariance matrices
              (estimation problem)
SYSOUT   - discrete output system description
NAMOUT   - name of section within SYSOUT
SYSIN    - continuous input system description
NAMIN    - name of section within SYSIN
```

Function

Depending on the switch SW, the system matrices and the matrices of the standard loss function or the covariance function is read from the continuous time state space system representation on the right hand side. The resulting matrices for the loss function or the covariance function are output according to the discrete time state space system representation on the left hand side.

The command uses the U input or the Y output.

The global variable NITER. is used as the maximum number of terms in the series expansions, while REPS. is used in the convergence test.

Method

Consider the continuous linear time-invariant system

$$\dot{x}(t) = Ax(t) + Bu(t)$$

and the problem of minimizing the functional

$$V = \int^{\infty} \begin{bmatrix} x \\ u \end{bmatrix}^T Q \begin{bmatrix} x \\ u \end{bmatrix} ds$$

where

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}, \quad Q_{21} = Q_{12}^{T}$$

Now assume that the input u(t) is piecewise constant over time intervals of length τ, the sample interval. Then

$$x(t+\tau) = \Phi x(t) + \Gamma u(t)$$

and

$$\tilde{V} = \sum_{0}^{\infty} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^{T} \tilde{Q} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}$$

where

$$\Phi = e^{A\tau}$$

$$\Gamma = B \int_{0}^{\tau} e^{As}\, ds$$

$$\tilde{Q} = \begin{bmatrix} \tilde{Q}_{11} & \tilde{Q}_{12} \\ \tilde{Q}_{21} & \tilde{Q}_{22} \end{bmatrix}$$

$$\tilde{Q}_{11} = \int_{0}^{\tau} e^{A^{T}s}\, Q_{11}\, e^{As}\, ds$$

$$\tilde{Q}_{12} = \int_{0}^{\tau} e^{A^{T}s} \{Q_{11} G(s) + Q_{12}\}\, ds$$

$$\tilde{Q}_{21} = \tilde{Q}_{12}^{T}$$

$$\tilde{Q}_{22} = \int_0^{\tau} \left\{ G^T(s) \, \{Q_{11} G(s) + Q_{12}\} + Q_{21} G(s) + Q_{22} \right\} ds$$

G(s) is given by

$$G(s) = \int_0^{s} e^{At} \, Bdt$$

It can be shown that

$$\tilde{Q}_{11} = P^T Q_{11} \Phi - SA$$

$$\tilde{Q}_{12} = S^T B + P^T Q_{12}$$

$$\tilde{Q}_{21} = \tilde{Q}^T$$

$$\tilde{Q}_{22} = B^T YB + B^T T^T Q_{12} + Q_{12}^T TB + Q_{22} \tau$$

where

$$\Phi = I + AP$$

$$P = I\tau + AT$$

$$S = YA + T^T Q_{11}$$

$$T = \tau^2 \int_1^{\infty} T_n$$

$$Y = \tau^3 \int_1^{\infty} Y_n$$

and

$$T_n = \frac{A\tau T_{n-1}}{}$$

$(n+1)!$

$$Y_n = \frac{1}{n+2} (Y_{n-1} A\tau + (A\tau)^T Y_{n-1} + Q_{11} T_n + T_n^T Q_{11}$$

Reference

K. Martensson: Linear quadratic control package Part II - The discrete problem, Report TFRT-3010 (1969), Dept. of Automatic Control, Lund Institute of Technology, Sweden.

Cautions, Restrictions

a) Even if the mixed term Q_{12} of the continuous problem is zero, the discrete problem will generally have a nonzero mixed term.

b) Note that TRANS does not change or reference the system matrices of the output system SYSOUT.

Hints

The matrices of discrete time state space system are computed from those of a continuous time representation in the command SAMP.

FEEDF

Purpose

To design a state space feedforward controller. Both discrete time and continuous time problems are handled.

Command

```
FEEDF1 SD < SM[(NAMEM)] SN[(NAMEN)]

SD       - system used to design a feedback
SM       - system modelling desired behaviour
NAMEM    - section within SM
SN       - system to receive feedforward and regulation
NAMEN    - section within SN
```

```
FEEDF2 [(SW)] SFF < SM[(NAMEM)] SN[(NAMEN)] LSD

SW       - switch selecting the Z output in SFF
           XN  - the nominal state
           ZN  - the nominal controlled output (default)
SFF      - system for the feedforward loop
SM       - same as in call to FEEDF1
NAMEM    - same as in call to FEEDF1
SN       - same as in call to FEEDF1
NAMEN    - same as in call to FEEDF1
LSD      - state feedback designed for SD
```

Function

There are two feedforward design commands solving two steps in the design process. The first, FEEDF1, takes as inputs a system modelling the desired behaviour of the controlled system and the controlled system itself. The output is a system file prepared for a state feedback design.

The user is then expected to design a state feedback for the system DS, such that the controlled system follows the model.

The third step is then to rearrange the given systems together with the state feedback to form a system suitable as a feedforward controller.

Both FEEDF1 and FEEDF2 will use the U input and the Z output, and this will be the form of SD. If SN includes a W input, it will be included in SFF as well. SFF will include a Y output, the generated feedforward control signal, and a Z output, either the nominal controlled output or the nominal state depending on the switch.

Method

The problem addressed in FEEDF has the following background, see Figure 1.

The system S is controlled both through a feedforward loop and a feedback loop. The system state is estimated ($\hat{x}$) in a state estimator system S_E. A feedforward controller S_{FF} computes the nominal control u_N and the nominal state x_N from the reference value u_r. The regulator system S_R compares x_N and $\hat{x}$ to form the regulator control signal u_R which is added to the nominal control u_N to form the total control signal u.

The feedforward system SFF will include a nominal model of S, called S_N.

In the description of the method, the continuous time form will be used.

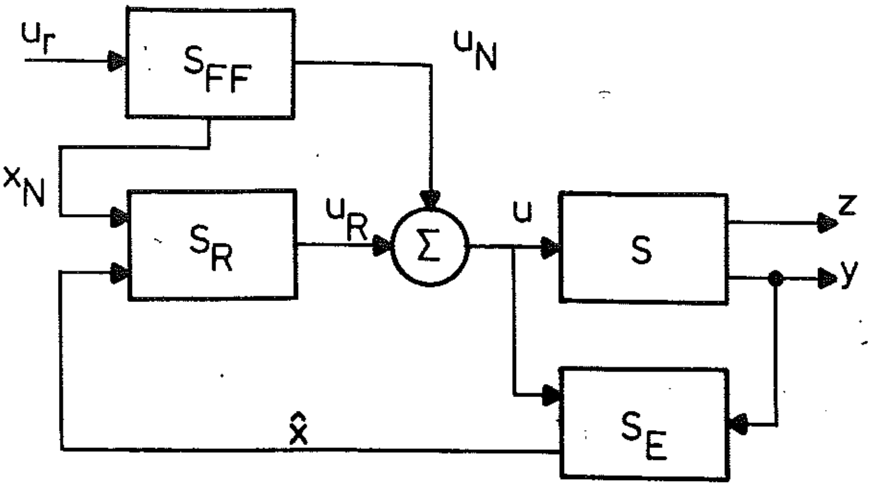


Figure 1. A system S with feedforward and feedback control.

The desired behaviour of the controlled system to <u>step</u> changes in the reference signal is modelled in SM:

$$\dot{x}_M = A_M x_M + B_M u_r$$

$$z_M = G_M x_M$$

The controlled system is given by the nominal model SN.

$$\dot{x}_N = A_N x_N + B_N u_N$$

$$z_N = G_N x_N$$

We finally model u_r as $\dot{u}_r = 0$.

The system SD is then formed:

$$\begin{bmatrix} \dot{x}_N \\ \dot{x}_M \\ \dot{u}_r \end{bmatrix} = \begin{bmatrix} A_N & 0 & 0 \\ 0 & A_M & B_M \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_N \\ x_M \\ u_r \end{bmatrix} + \begin{bmatrix} B_N \\ 0 \\ 0 \end{bmatrix} u_N$$

$$Z_{SD} = \begin{bmatrix} -G_N & G_M & 0 \end{bmatrix}$$

The system file SD will also include a standard and extended loss function definition. The extended loss function matrices are generated as

$$EQ_1 = 0 \qquad EQ_2 = I \qquad EQ_4 = I \qquad EQ_5 = 0$$

In the next step (the second), the user should design a state feedback for this system such that Z_{SD} stays small.

In the third step, FEEDF2 is used to form the feedforward controller. The L designed in the second step is partitioned:

$$u_N = \begin{bmatrix} -L_N & -L_M & -L_r \end{bmatrix} \begin{bmatrix} x_N \\ x_M \\ u_r \end{bmatrix}$$

The feedforward controller is then the dynamical system:

$$\begin{bmatrix} \dot{x}_N \\ \dot{x}_M \end{bmatrix} = \begin{bmatrix} A_N - B_N L_N & -B_N L_M \\ 0 & A_M \end{bmatrix} \begin{bmatrix} x_N \\ x_M \end{bmatrix} + \begin{bmatrix} -B_N L_r \\ B_M \end{bmatrix} u_r$$

$$\{ u_N = \} \; y = \begin{bmatrix} -L_N & -L_M \end{bmatrix} \begin{bmatrix} x_N \\ x_M \end{bmatrix} + \begin{bmatrix} -L_r \end{bmatrix} u_r$$

$$\{ x_N = \} \; z = [\; I \quad 0 \;] \begin{bmatrix} x_N \\ x_M \end{bmatrix} \qquad \text{if} \quad sw = XN$$

$$\{ z_N = \} \; z = \begin{bmatrix} G_N & 0 \end{bmatrix} \begin{bmatrix} x_N \\ x_M \end{bmatrix} \qquad \text{if} \quad sw = ZN$$

Cautions, Restrictions

The design of the state feedback in the second step is intended to be performed using PENLT & OPTFB. The system S_M should be stable and S_N controllable and EQ_4 should have full rank, all very reasonable requirements. Then the system SD will be stabilizable and detectable through the loss function.

The designer of the state feedback L is warned not to:

a) Try to change eigenvalues belonging to S_M or to the input signal model (a set of integrators) since they obviously belong to non-controllable modes.

b) Introduce a penalty directly on the state variables, either through EQ1 or EQ3. This would violate the entire idea of the method which is to give the loss function a special form such that the error $Z_M - Z_N$ is minimized.

Hints

a) FEEDF1 is used once to rearrange the systems defining the problem.

b) PENLT & OPTFB are used iteratively to reach an L giving satisfactory performance.

c) FEEDF2 with sw = ZN is intended to be used in the iteration loop b) to form a system with the feedforward control and the nominal controlled output as results. This system may be simulated to evaluate the current L in the design loop.

d) FEEDF2 with sw = XN is used finally to give the feedback controller with nominal control U_N and nominal state X_N as Y and Z outputs.

e) The choice of dynamics for the model system S_M should reflect the practical limitations on the achievable performance from S_N. If it is difficult to obtain good agreement between Z_N and Z_M, the choice of S_M should be reconsidered.

KALFI

Purpose

To design a state reconstructor gain matrix for a linear state space system, minimizing the reconstruction error covariance. Both continuous time and discrete time systems are allowed.

Commands

```
KALFI [(INI)] [/['K'] ['K1'] ['P']/] [K] [K1] [P] < SYST[(NAME)]

INI      - initialization switch = 'AUTO'/'MAN'
           AUTO - automatic initialization of P
           MAN  - P is initialized from R0
K        - filter gain matrix given measurements up to
           the previous time
K1       - filter gain matrix given measurements including
           the current time
P        - solution to the stationary Riccati equation
SYST     - system file name
NAME     - section name within SYST
```

Subcommands (available only in case of slow convergence)

```
CONT     - iterate NITER. more iterations
LOOK     - write P on the the terminal
KILL     - exit immediately
X        - write partial results and exit
```

Function

The right hand side system description is inspected and the system matrices A and C are read. Note that only the output y is considered, being the outputs available as measurements. The system description also specifies the covariance matrices for the state noise v and the measurement noise e. The matrices B_v and D_e are assumed unity.

Then the stationary Riccati equation is solved. Depending on the output file identification flags, the filter gain matrices K and/or K1 and the stationary Riccati solution P are output. The distinction between K and K1 is given below.

Refer to the command OPTFB for a discussion of the argument INI and the global variables used.

Method

In the continuous time case, assume that the system is given by (the blocking of the inputs and the possible inclusion of a D-matrix is immaterial to KALFI and is left out):

$$\begin{cases} \dot{x} = Ax + Bu + v & E\, v^T v = R_1 \\ y = Cx & E\, e^T e = R_2 \end{cases}$$

The Kalman filter reconstruction of x is then given by

$$\dot{\hat{x}} = A\hat{x} + Bu + K\,(y - C\hat{x})$$

where K is given by

$$K = PC^T R_2^{-1}$$

and P is the solution to the Riccati equation

$$\dot{P} = AP + PA^T + R_1 - PC^T\, R_2^{-1}\, CP.$$

Here we have assumed $E\, v^T e = R_{12}$ to be zero. This is not necessary as the equations used in KALFI allows a non-zero R_{12}.

In the discrete time case we have similarly

$$x(t+1) = Ax(t) + Bu(t) + v$$

$$y(t) \quad = Cx(t) \qquad\qquad + e$$

Now the state reconstruction will depend on the time instance of the last available measurement. We have two possibilities:

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K_1 [y(t) - C\hat{x}(t|t-1)]$$

$$\hat{x}(t+1|t) = A\hat{x}(t|t) + Bu(t) =$$

$$= A\hat{x}(t|t-1) + Bu(t) + K[y(t) - C\hat{x}(t|t-1)]$$

where

$$K_1 = PC^T(CPC^T + R_2)^{-1}$$

$$K = A \cdot K_1$$

and P is the stationary solution to

$$P(t+1) = AP(t)A^T + R_1 - AP(t)C^T[CP(t)C^T + R_2]^{-1}CP(t)A^T$$

R_1, R_{12}, and R_2 have the same significance as above. Also in this case R_{12} may be included although not shown in the equation above.

For the method of solving the Riccati equation, refer to the command OPTFB.

References

K.J. Astrom: Introduction to Stochastic Control Theory. Academic Press, 1970.

Hints

a) The covariance matrices R_1, R_2, and maybe R_{12} reflect the uncertainty in the system description and the measurements. They can also be used to obtain a desired dynamic response of the state reconstructor in a fashion similar to the use of OPTFB.

b) The actual state reconstructor (Kalman filter) is formed using SYSOP, cf. the example below.

c) KALFI results in a full order observer. Reduced order observers may be designed using KALFI with the aid of LUEN.

d) Direct pole placement may be possible e.g. RECON.

<u>Examples</u>

The system S is given:

```
BEGIN
CONTINUOUS STATE SPACE REPRESENTATION
DYNAMICS, AGGREGATE: S,
DX/DT = A*X + B*U + BV*V
Y = C*X + DE*E
COVARIANCE FUNCTION, AGGREGATE: SC,
R1: R1,   R2: R2
END
```

The following numeric values are used:

$$S{:}A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad S{:}B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad S{:}B_V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$S{:}C = [\ 1 \quad 0\ ] \qquad S{:}DE = [\ 1\ ]$$

$$SC{:}R_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad SC{:}R_2 = [\ 1\ ]$$

Then the following series of commmands are used, first to find the filter gain K, then to construct the Kalman filter:

```
>KALFI /K/ K < S
>SYSOP SF < S /M/ K
   >LOOK
   >IN U1 < U
   >IN V1 < Y2
   >IN E1 < 0
   >IN U2 < W - Y1
   >OUT Y < X1
   >X
>LIST (T) SF
```

The output from the LOOK subcommand is:

```
                       S            S          K
                       (       )   (       )
CONTROL     INPUTS                 U1     1    U2    1
STOCHASTIC  INPUTS                 V1     2
MEASUREMENT ERRORS                 E1     1
            STATES     X     2     X1     2
MEASURED    OUTPUTS                Y1     1    Y2    2
```

The generated system description is:

Design
KALFI

```
BEGIN
"SYSOP SF<S /M/ K
"
CONTINUOUS STATE SPACE REPRESENTATION
DYNAMICS, AGGREGATE: SF,
DX/DT = A*X + B*U + BW*W
Y = C*X
END
```

The obtained values of the matrices are:

$$K = \begin{bmatrix} 1.73205 \\ 1.0 \end{bmatrix}$$

$$SF{:}A = \begin{bmatrix} -1.73205 & 1.0 \\ -1.0 & 0 \end{bmatrix} \qquad SF{:}B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$SF{:}B_w = \begin{bmatrix} 1.73205 \\ 1.0 \end{bmatrix} \qquad SF{:}C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The U-input is the input to the system while the W-input is the measured output from the system. The Y-output is the reconstructed state.

LUEN

Purpose

To design a reduced order observer (a Luenberger observer).

Command

```
LUEN1  T SYST1 < SYST2[(NAME2)] [EPS]

T     - name of transformation matrix
SYST1 - system description file name for the new (reduced)
         system
SYST2 - system description file name for the original
        system
NAME2 - section name of the original system, syst2
EPS   - test quantity
          by default eps = the reserved variable reps.


LUEN2  SYST1 < SYST2[(NAME2)]  T  K  [EPS]

SYST1 - system description file name for the luenberger observer
        (of reduced order)
SYST2 - system description file name for the original system
        (of full order)
NAME2 - section name of the original system, SYST2
T     - name of transformation matrix (full order), given from LUEN1
K     - name of gain matrix (reduced order), given from pole placement
EPS   - test quantity for matrix inversion
        by default EPS = the reserved variable REPS.
```

Function

There are two Luenberger observer design commands solving two steps in the design process. The first, LUEN1, takes as input the system for which an observer is desired. LUEN1 computes a transformation separating the states directly available from the measurement y from those that are not. The matrices of the latter part of the system is output together with the transformation matrix.

The user is then expected to design a full order observer for the not directly measured states.

Finally, LUEN2 is applied to compute the matrices of a system representation for the desired reduced order (Luenberger) observer. The input to LUEN2 is the original system, the transformation found by LUEN1 and the full order state reconstructs gain matrix found by the user.

Method

The observer design is divided into three steps. The system for which we intend to design an observer is given (here we use a continuous time description, the same applies to discrete time):

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases}$$

First a coordinate transformation $\xi = T_1 x$ is designed giving the system representation:

$$\begin{cases} \dot{\xi} = T_1 A T_1^{-1} \xi + T_1 Bu \\ y = C T_1^{-1} \xi \end{cases}$$

such that $CT_1^{-1} = [\ 0 \quad I\ ]$

Thus ξ may be partitioned into ξ_1 and ξ_2, where ξ_2 are the states directly measured through y.

$$\begin{cases} \begin{bmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \\ y = [\ 0 \quad I\ ] \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix} = \xi_2 \end{cases}$$

The output of the first step as implemented in LUEN1 is thus the following system and the transformation T_1.

$$\begin{cases} \dot{\xi}_1 = A_{11}\ \xi_1 + B_1 u \\ y_1 = A_{21}\ \xi_1 \end{cases}$$

In step 2 of the design, the user is required to compute a full order observer for ξ_1, the not directly measured states of the original system. Specifically, a K is required so that

$$A_{11} - K\cdot A_{21}$$

is stable with suitable eigenvalues. This can e.g. be done in KALFI.

In step 3, finally, the one performed in LUEN2, a transformation $\xi' = T_2 T_1 x$ is performed on the original system, where

$$T_2 = \begin{bmatrix} I & -K \\ 0 & I \end{bmatrix}$$

with the same partitioning as above, we have

$$\begin{bmatrix} \dot{\xi}'_1 \\ \dot{\xi}'_2 \end{bmatrix} = \begin{bmatrix} A'_{11} & A'_{12} \\ A'_{21} & A'_{22} \end{bmatrix} \begin{bmatrix} \xi'_1 \\ \xi'_2 \end{bmatrix} + \begin{bmatrix} B'_1 \\ B'_2 \end{bmatrix} u$$

The importance with the special choice of T_2 is that $A'_{11} = A_{11} - K\cdot A_{21}$, designed to be nice, and that still $\xi'_2 = \xi_2 = y$. Introducing two new matrices C' and D'_w and noting that the desired state estimate is

$$\hat{x} = (T_1\ T_2)^{-1} \begin{bmatrix} \xi'_1 \\ \xi'_2 \end{bmatrix} \triangleq C'\cdot\xi'_1 + D'\cdot\xi'_2$$

we have

$$\dot{\xi}'_1 = A'_{11}\ \xi'_1 + B'_1 u + A'_{12} y$$

$$\hat{x} = C'\xi'_1 + D'_w y$$

This is the result from LUEN2. $\hat{x}$ is returned as the Y output, u is expected as the U input while y is expected as the W input.

Examples

As example we use the one-dimensional particle with the position as the output y. Thus we have the system SD:

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

with

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \text{and} \quad C = [\ 1 \quad 0\ ].$$

we then do:

```
>LET DELTA. = 0.                "Continous time
>SYST S1                        "A standard state space system
>LUEN1 T S1 < SD
>LIST T
```

The transformation T obtained as well as the A, B, and C matrices of S1 are shown:

$$T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad A_{S1} = [\ 0\ ], \quad B_{S1} = [\ 1\ ], \quad C_{S1} = [\ 1\ ].$$

We now choose K = [1] as the value of the required filter gain matrix. This gives a stable reconstructor since

$$A_{S1} - K \cdot C_{S1} = [-1].$$

The Luenberger observer is then found through

```
>SYST (SC) SOBS
   >INS BW
   >INS DW
   >X
>LUEN2 SOBS < SD T K
```

The observer has the state equation

$$\dot{x} = Ax + Bu + B_w W$$

$$y = Cx + D_w W$$

with

Design
LUEN

$$A = [\ -1\ ] \qquad B = [\ 1\ ] \qquad B_w = [\ -1\ ]$$

$$C = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad D_w = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Referring to the original system we find:

$$\begin{cases} \hat{x}_1 = y \\ \hat{x}_2 = \xi + y \end{cases}$$

where $\dot{\xi} = - \xi + u - y$

Note that the state variable of the reconstructor has no direct physical significance. This is the price one has to pay for the lower observer order, compare the example in RECON.

OPTFB

Purpose

To design a state feedback matrix for a linear state space system, minimizing a quadratic loss function. Both continuous time and discrete time systems are allowed.

Commands

```
OPTFB [(INI)] [/['L'] ['S']/] [L] [S] < SYST[(NAME)] [ALPHA]

INI      - initialization switch = 'AUTO'/'MAN'
           AUTO - automatic initialization of S
           MAN  - S is initialized from Q0
L        - feedback matrix
S        - solution to the stationary Riccati equation
SYST     - system file name
NAME     - section name within SYST
ALPHA    - stability coefficient, i.e. the system matrix A is
           replaced by (A + ALPHA*I) (default: 0)
```

Subcommands (available only in case of slow convergence)

```
CONT     - iterate NITER. more iterations
LOOK     - write S on the the terminal
KILL     - exit immediately
X        - write partial results and exit
```

Function

The right hand side system description is inspected and the system matrices A and B are read. Note that only the block B_u is used, u being the inputs available for control purposes. The system description also specifies the matrices of the loss function. Q_1 and Q_2 should be included, Q_0 and Q_{12} are optional. Then the stationary Riccati equation is solved. Depending on the output file identification flags, the state feedback matrix L and/or the stationary Riccati solution S are output.

Refer to the next section for a discussion on the arguments INI and ALPHA. OPTFP references some global variables. REPS. is used as test quantity in relative tests in numerical operations (solving matrix equations) while CEPS. is used as convergence criterion in the solution of the Riccati equation. A maximum of NITER. iterations are allowed. PRINT. is used to control line printer listing according to:

= 0   no output.

> 0, ≤ 100   the number of iterations is printed on the terminal.

> 100   S is listed on the line printer every PRINT.-100 iterations.

Method

Given the system equation

$$\dot{x} = Ax + Bu \quad \text{or} \quad x(t+1) = Ax(t) + Bu(t)$$

and the quadratic criterion function

$$J = \int (x^T Q_1 x + 2x^T Q_{12} u + u^T Q_2 u)\, dt$$

or

$$J = \sum_t (x^T Q_1 x + 2x^T Q_{12} u + u^T Q_2 u)$$

the optional (stationary) state feedback is known to be given by

$$u = - Lx$$

where L is given in the continuous time case or the discrete time case by the respective equations:

$$L = Q_2^{-1}(Q_{12}^T + B^T S) \quad \text{or} \quad L = (Q_2 + B^T SB)^{-1}(Q_{12}^T + B^T SA)$$

In both cases, S is the stationary solution to the Riccati equation

$$-\dot{S} = A^T S + SA + Q_1 - (SB + Q_{12})\, Q_2^{-1}\, (Q_R^T + B^T S)$$

or

$$S(t-1) = A^T S(t) A + Q_1 -$$

$$- (A^T S(t)\, B + Q_{12})\, (Q_2 + B^T S(t)\, B)^{-1}\, (Q_{12}^T + B^T S(t)\, A)$$

A stationary solution to the Riccati differential or difference equation is found through iteration in time, with a test for convergence, specifically the norm of the difference between two succesive values is required to be less than CEPS.. In the continuous time case, the time step is chosen based on the eigenvalues, for the discrete time case, the sample interval is used.

If INIT = AUTO, the initial value for S is computed using eigenvalues / eigenvectors of the Euler matrix for the problem, see Reference [1]. This ensures convergence in very few steps for most problems. If this method, originally due to Potter fails, Q_0 is used as the initial value. If Q_0 is absent, 100*I is used.

If INI = MAN, Q_0 is always used as initial guess.

If the argument ALPHA is given in the command line, the matrix A in the system description is for a continuous time problem replaced by A + $\alpha\cdot$I. This corresponds to placing a weighting factor in the continuous time criterion (see Reference [2]):

$$J^* = \int e^{2\alpha t} \{ x^T Q_1 x + 2x^T Q_{12} u + u^T Q_2 u \} \ dt.$$

It is obvious that this little trick ensures that a stabilizing feedback designed in this manner will result in a closed loop system with poles to the left of $-\alpha$.

References

[1] K. Martensson: On the matrix Riccati equation. Report TFRT-3020, Dept. of Automatic Control, Lund Institute of Technology, Sweden, 1970.

[2] B.D.O. Anderson: J.B. Moore: Linear Optimal Control. Prentice Hall, 1971.

Hints

a) Note that although the loss function in some cases may be derived from physical considerations, the normal use is to use the loss function matrix elements (mainly the diagonal ones) as knobs to twist in order to achieve a suitable performance.

b) As the design based on OPTFB typically is iterative, it is often natural to use it in a macro together with some performance evaluating commands.

c) The performance may be studied through simulation (SIMU), eigenvalue computation (EIGEN, POLES) or frequency responses (SPSS).

d) The command PENLT may give a set of knobs to twist attached with more intuition, i.e. the loss function terms have a different interpretation.

e) Some results in linear quadratic theory assumes Q_1 and Q_2 to be positive definite and Q_{12} to be zero. These assumptions are <u>not</u> necessary in order to apply OPTFB. PENLT will assume Q_{12} to be included.

f) The closed loop system is continuated using SYSOP.

g) OPTFB assumes that the state is available for feedback. If this is not so in practice, it may be reconstructed, cf. LUEN, KALFI etc. A state feedback may be converted to an output feedback in REDFB.

h) Direct pole placement may be possible, e.g. PPLAC.

<u>Examples</u>

As an example, we will use a state space model of the ball and beam process (*).

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \\ z = Gx \end{cases}$$

with

$$A = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad G = [\,0 \quad 0 \quad 0 \quad 1\,]$$

The initial state is

(*) J. Wieslander: Interaction in Computer Aided Analysis and Design of Control Systems, Chapter 7. Thesis TFRT-1019, Dept. of Automatic Control, Lund Institute of Technology, Sweden, 1979.

$$x0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0.5 \end{bmatrix}$$

The two measurements are the beam angle and the ball position, y_1 resp. y_2. The single controlled variable is the ball position.

The command PENLT and the extended loss function will be used here for demonstration purpose. We will use it to introduce a penalty on the controlled output alone (and on the input, naturally). The advantage is that we need not bother with the interpretations of the state variables, although they of course are known here as in most other cases.

The system is described in the system file BAB. The matrices of the extended loss function are found in the aggregate BABE. Other matrices are invisible in this example.

The following macro is used:

```
MACRO ITER ALPHA
PENLT BAB                       "Convent loss function
OPTFB L < BAB ALPHA             "Compute state feedback
DELETE (T) SCL                  "In case SCL was present
SYSOP SCL < BAB /M/ L
IN U1 < -Y2+U
IN U2 < X1
OUT Y < Y1                      "The measured outputs
OUT Z < -Y2                     "The control signal
X
POLES < SCL                     "Display closed loop poles
SUSPEND                         "Have time to inspect them
SIMU /Y Z/ Y U < SCL ZERO
PLOT Y / U
END
```

A linear quadratic state feedback is then designed with the following commands. Refer to the indicated figures. Initially the system BAB with the appropriate system matrices have been entered. The initial value of the extended loss function is:

BABE: EQ_1 = 0 BABE: EQ_2 = [1]

BABE: EQ_4 = [1] BABE: EQ_5 = [0]

```
>LET TICK. = 0.1
```

```
>INSI ZERO  100  0.1           "A sample interval for SIMU
   >ZERO                       "Define a zero input
   >X
>HCOPY ON
>ITER 0.
   >HCOPY                      "Copy figure 1
   >RESUME
>HCOPY                         "Copy figure 2
>ALTER BABE: EQ4(1 1) 10.
>ITER 0.
   >HCOPY                      "Copy figure 3
   >RESUME
>HCOPY                         "Copy figure 4
>ALTER BABE: EQ4(1 1) 100
>ITER 0.
   >HCOPY                      "Copy figure 5
   >RESUME
>HCOPY                         "Copy figure 6
>ALTER BABE: EQ5(1 1) 50
>ITER 0.
   >HCOPY                      "Copy figure 7
   >RESUME
>HCOPY                         "Copy figure 8
>LIST L
```

The L designed so far has the value

```
     L  =  [ 4.10513   12.5312   17.3385   10.000 ]
```

Of course, the indicated series of interactions is not necessarily the best concievable. It is however important to note that:

a) Increasing the penalty on the output variable Z from 1 through 10 to 100 results in a shift of poles to the left in the complex plane and an increasing speed of response.

b) Increasing the penalty on the derivative of the output results in an increased damping as can be noted both in the pole location and in the time responses.

An alternate method of achieving a desired speed of response without impaired damping is to use the argument ALPHA in the command OPTFB. This is illustrated below.

```
>ALTER BABE: EQ4(1 1) 10
>ZEROM BABE: EQ5 1             "Revert to figures 3 & 4
>ITER 0.5
   >HCOPY                      "Copy figure 9
   >RESUME
>HCOPY                         "Copy figure 10
>LIST L
```

This L is chosen as the final design, and is used in the example found in REDFB.

L = [4.73347 13.5696 16.1716 8.51044]

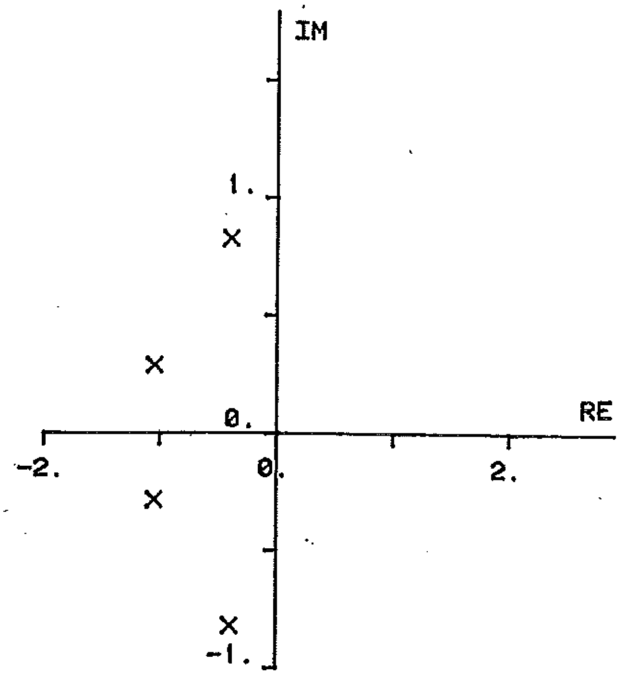


Figure_1

Design
OPTFB

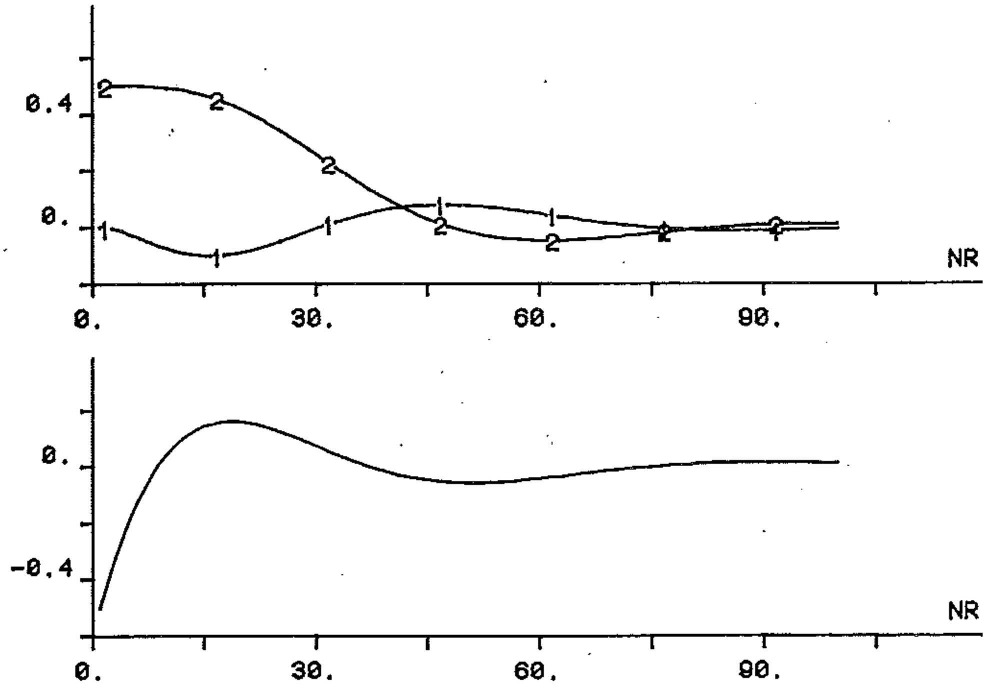


Figure 2

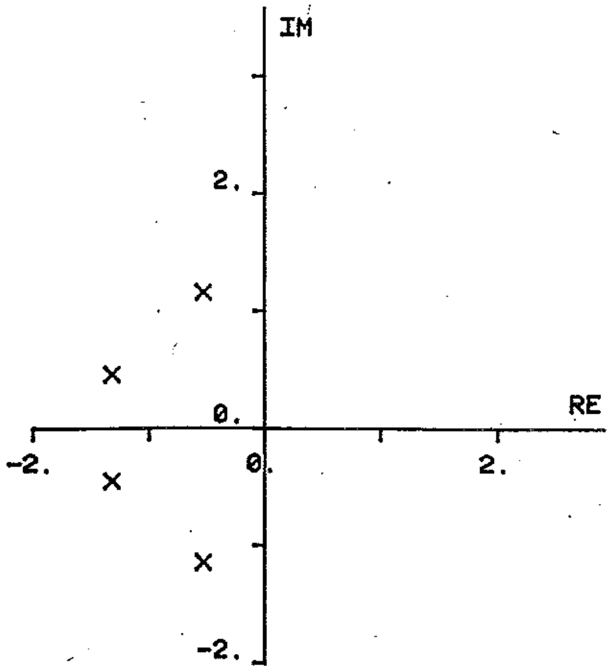


Figure 3

Design
OPTFB

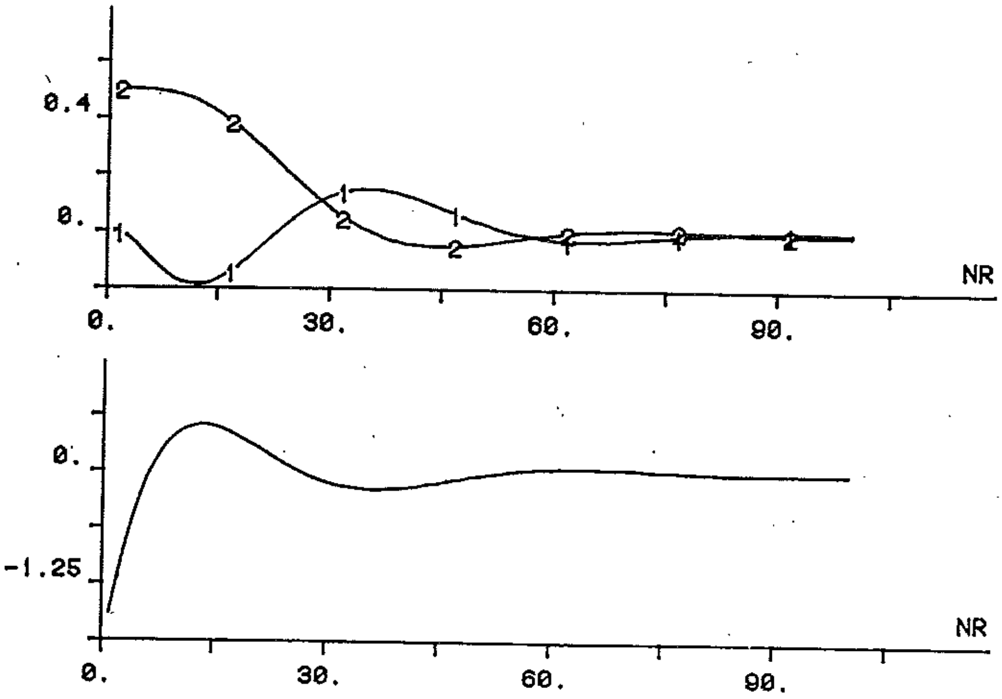


Figure_4

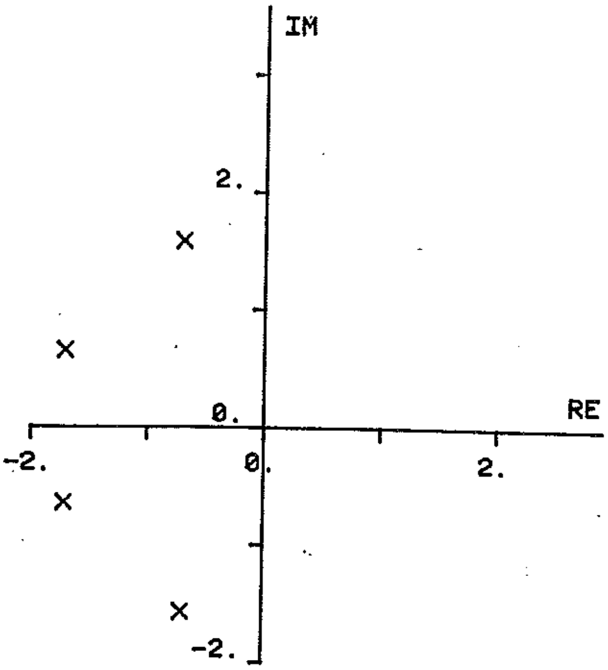


Figure_5

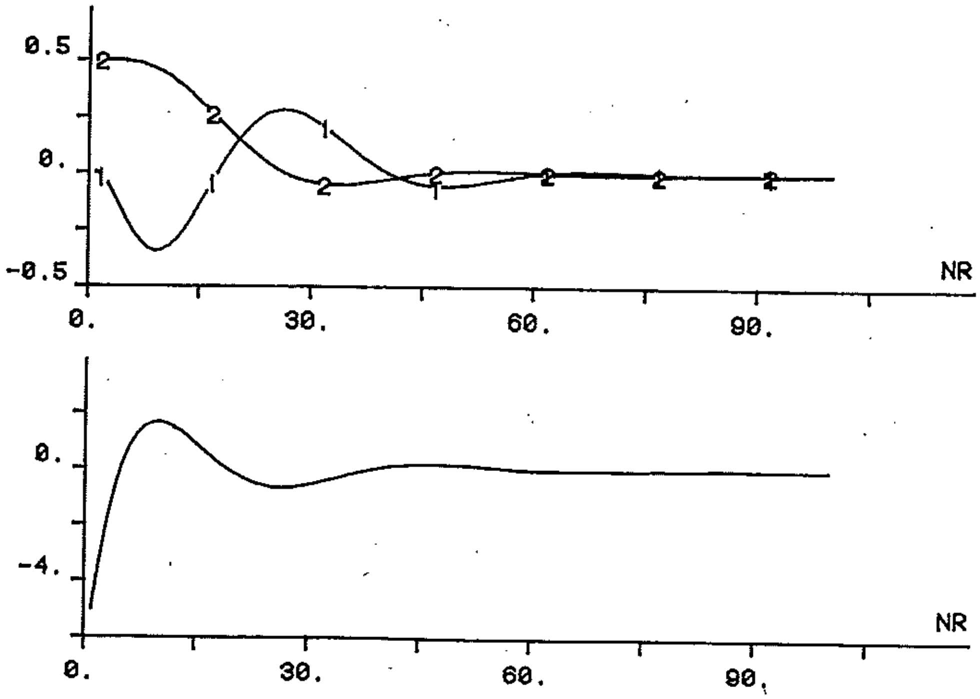


Figure 6

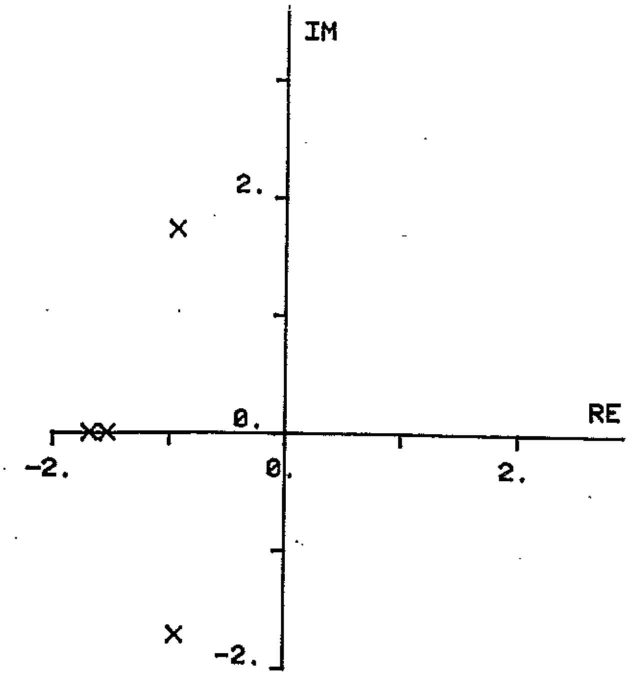


Figure 7

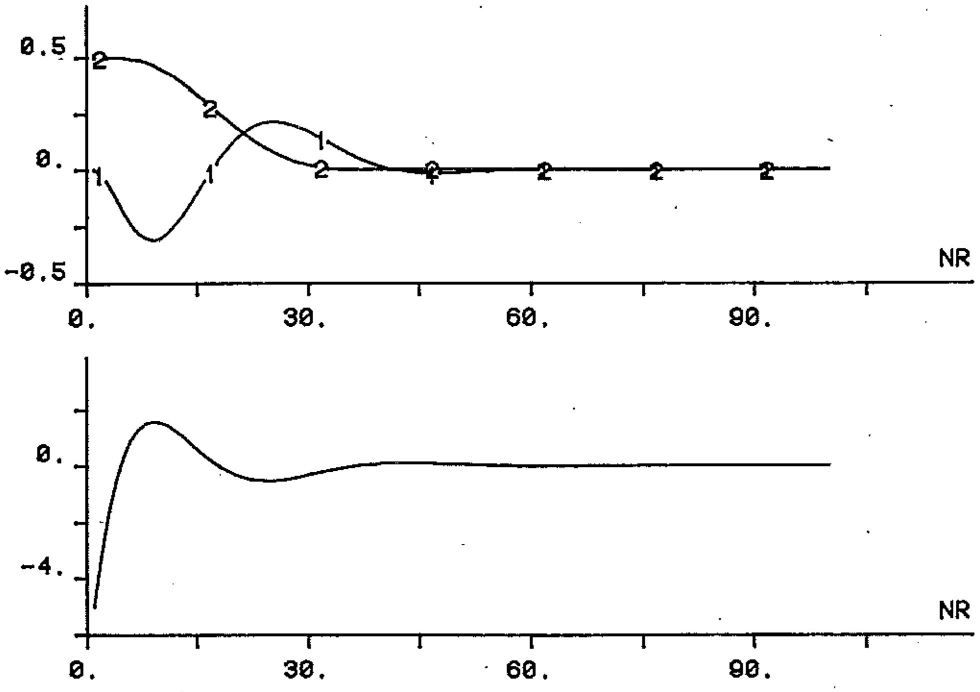


Figure 8

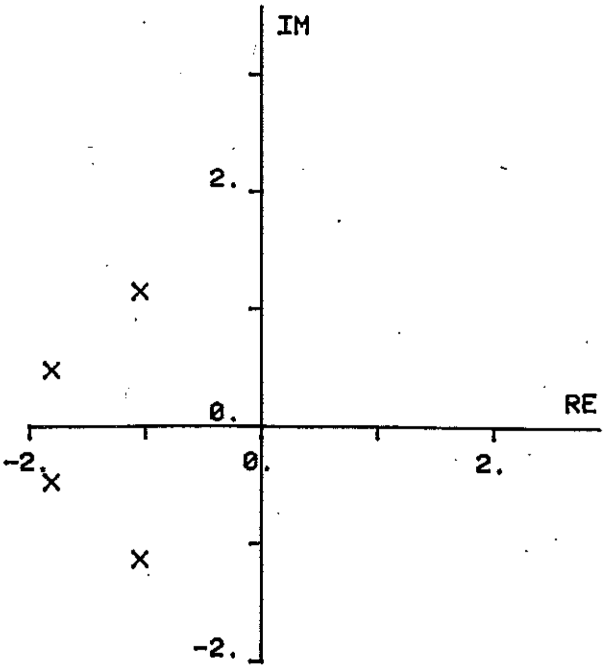


Figure 9

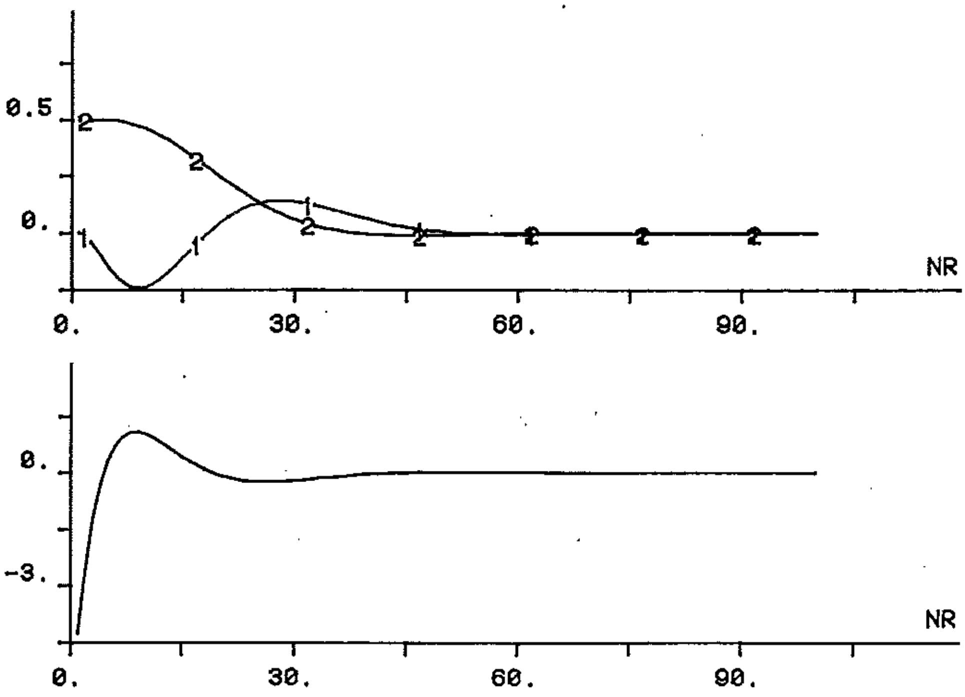


Figure 10

PENLT

Purpose

To reduce the extended loss function of a state space system to the standard loss function handled by OPTFB. Both the continuous and the discrete time problems can be handled.

Command

```
PENLT SYS[(NAME)] [AMOD]

SYS      - system file name
NAME     - name of section within SYS
AMOD     - model for closed loop dynamics, i.e. the state
           derivative DX/DT is replaced by (DX/DT - AMOD*X)
           (default: 0)
```

Function

The specified state space system representation is inspected and the matrices of the system and the extended loss function are read. The loss function is then transformed into its standard form and its construct matrices are output.

Note that PENLT will handle the input U (the control input) and the output Z (the controlled output). The optional argument AMOD is the desired A-matrix of the closed loop system, cf. below and hint b).

Method

The extended loss function used in PENLT has the following integrand (for discrete time: summand):

$$x^TQ_{e1}x + 2x^TQ_{e12}u + u^TQ_{e2}u + (\dot{x}-\bar{A}x)^TQ_{e3}(\dot{x}-\bar{A}x) +$$

$$+ z^TQ_{e4}z + z^TQ_{e5}z$$

In the discrete time case, the derivatives are approximated by a difference:

$$\dot{V} \approx [V(t+T) - V(t)] / T$$

In practical work with the quadratic loss function used in OPTFB, of-diagonal elements in Q_1 and Q_2 are seldom used and Q_{12} is kept zero. The motivation for the extended loss function is then that although the user still restricts himself to specifying diagonal elements, more freedom and intuition is available.

a) The term $z^TQ_{e4}z$ allows direct specification of a penalty on large values of the output.

b) The terms $\dot{x}^TQ_{e3}\dot{x}$ and $\dot{z}^TQ_{e5}z$ allows a penalty on too rapid changes of state variables and output variables.

c) The term $(\dot{x}-\bar{A}x)^TQ_{e3}(\dot{x}-\bar{A}x)$ will include the closed loop system to behave like $\dot{x} = \bar{A}x$, i.e. $\bar{A}$ (AMOD in the arguments) specifies derived eigenvalues and eigenvectors of the closed loop system.

The xtended loss function is converted to the standard form according to the following formulae:

$$Q_1 = Q_{e1} + (A-\bar{A})^TQ_{e3}(A-\bar{A}) + G^TQ_{e4}G + A^TG^TQ_{e5}GA$$

$$Q_{12} = Q_{e12} + BQ_{e3}(A-\bar{A}) + GQ_{e4}H + A^TG^TQ_{e5}GB$$

$$Q_2 = Q_{e2} + B^TQ_{e3}B + H^TQ_{e4}H + B^TG^TQ_{e5}GB$$

Cautions, Restrictions

a) Note that if the matrix H is non-zero, i.e. there is a direct term from control inputs to controlled outputs, then Q_{e5} must be zero.

b) Note that Q_{12} will in general be non-zero.

Hints

a) If only diagonal elements in the extended loss function matrices are used and they are non-negative, and if Q_{12} is zero, then the standard loss function will be positive semi-definite.

b) If E is a matrix with desired eigenvectors and D = diag(λ_i) contains the corresponding desired eigenvalues, then

$\bar{A} = E \cdot D \cdot E^{-1}$.

However, the choice of desired eigenvalues and eigenvectors will demand a good process knowledge.

Examples

A detailed example where PENLT is used can be found in the description of the command OPTFB.

To illustrate the use of the argument AMOD the following somewhat synthetic example may be used. The following system (S) is given:

$$\dot{x} = Ax + Bu$$

with

$$A = \frac{1}{3}\begin{bmatrix} -2 & 1 & 2 & 4 \\ 1 & -5 & -1 & 7 \\ 2 & -1 & -2 & -4 \\ 0 & 0 & 0 & -9 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

A state feedback with u = -Lx will give a closed loop system with

$$A_{CL} = \begin{bmatrix} -0.667 & 0.333 & 0.667 & 1.333 \\ -0.333 & 0.667 & -4.667 & 7.667 \\ -2 & 0 & -3 & -1 \\ -2 & -2 & 2 & -8 \end{bmatrix}$$

if

$$L = \begin{bmatrix} 2.6667 & -0.3333 & 2.3333 & -0.3333 \\ 2 & 2 & -2 & 5 \end{bmatrix}$$

Now, introducing an extended loss function into the system description S with

$$EQ_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad EQ_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and issuing the following series of commands, we get:

```
>PENLT S ACL
>OPTFB LA < S
>LIST LA
```

$$LA = \begin{bmatrix} 1.891 & -0.4200 & 2.0371 & -0.7029 \\ 1.2514 & 1.5119 & -1.6598 & 3.7235 \end{bmatrix}$$

This is still far from the L above, but increasing EQ_3 we get the following results:

EQ_3 = 10*I

$$LA = \begin{bmatrix} 2.5425 & -0.3553 & 2.3002 & -0.4117 \\ 1.8800 & 1.9292 & -1.9559 & 4.8117 \end{bmatrix}$$

EQ_3 = 100*I

$$LA = \begin{bmatrix} 2.6534 & -0.3358 & 2.3300 & -0.3420 \\ 1.9872 & 1.9925 & -1.9954 & 4.9801 \end{bmatrix}$$

EQ_3 = 1000*I

$$LA = \begin{bmatrix} 2.6653 & -0.3336 & 2.3330 & -0.3342 \\ 1.9987 & 1.9992 & -1.9995 & 4.9980 \end{bmatrix}$$

The point is that if a desirable closed loop performance is known in terms of the closed loop A-matrix, a state feedback can be designed in the manner shown here.

PPLAC

Purpose

To design a state feedback such that the closed loop poles have desired values. The system must be represented in continuous time or discrete time form with a single output.

Command

```
PPLAC L [ [SYST1][(NAME1)] ] < SYST2[(NAME2)] EVAL [EPS]
```

L – name of feed-back matrix size 1*NX
SYST1 – system description file name for the closed loop system
by default SYST1 = SYST2 if NAME1 is present
NAME1 – section name of the new system, SYST1
SYST2 – system description file name for the original system (A,B,C,D)
NAME2 – section name of the original system, SYST2
EVAL – name of locus file containing desired poles
EPS – test quantity
by default EPS = the reserved varible REPS.

Function

The input system description matrices are read together with the desired closed loop pole locations given in EVAL. The state feedback L is computed optionally together with the matrices of the closed loop system. Note that PPLAC will use the control input U. The reserved global variable REPS. is used in the transformation to controllable canonical form.

Method

The given system (the equations for the discrete time case are analogous)

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

is transformed to controllable canonical form through $\xi = Tx$

$$\dot{\xi} = TAT^{-1}\xi + TB\,u = A_\xi \xi + B_\xi u$$

The form of the matrices A_ξ and B_ξ is such that the control

law

$$u = - L_{\xi} \xi = - Lx \qquad L = L_{\xi} T$$

is immediately obvious, see the reference, page 249.

The optionally output closed loop system is computed from $u = u_r - Lx$ and has thus the form:

$$\begin{cases} \dot{x} = (A-BL)\ x + Bu_r \\ y = Cx + Du \end{cases}$$

References

K.J. Astrom: Reglerteori. Almqwist & Wiksell, 1976.

Hints

The locus file EVAL may be given from POLES & PLEV, see the example below.

Examples

Assume that a DC-motor with transfer function

$$G(s) = \frac{1}{s(s + 0.5)}$$

is given and that it is desired to design a state feedback giving two complex conjugated poles with natural frequency 2 and damping 0.7.

The states are position x_2 and velocity x_1:

$$\dot{x} = \begin{bmatrix} -0.5 & 0 \\ 1 & 0 \end{bmatrix}$$

$$y = [\ 0 \quad 1\ ]\ x$$

The following sequence of commands is used:

```
>POLES  E < S
>PLEV  E1 < E
   >LOOK
   >ALT  1  -1.5  0.5  _ 2
   >LET  T. = 2./1.5811
   >SCALE  1  T.
   >DAMP  1  0.7
   >LOOK
   >X
>SYST SCL
>PPLAC  L  SCL < S  E1
>LLIST L
```

First the open loop poles are computed. They are then changed to the desired closed loop locations using PLEV. Note the use of EXAM - LET - SCALE to achieve the desired natural frequency. The result of the second LOOK is shown in Figure 1. Finally a closed loop system description is defined named SCL and the actual pole placement is done. The resulting L is

$$L = [\ 2.3 \quad -4.0\ ]$$

while the closed loop matrices are

$$SCL{:}A = \begin{pmatrix} -2.8 & -4.0 \\ 1.0 & 0 \end{pmatrix} \qquad SCL{:}B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$SCL{:}C = [\ 0 \quad 1\ ]$$

RECON

Purpose

To design a state reconstructor such that the reconstructor poles have desired values. The system must be represented in continuous time or discrete time form with a single output.

Command

```
RECON K [SYST1] < SYST2[(NAME2)] EVAL [EPS]
```

```
K      - name of gain matrix  size NX*1
SYST1  - system description file name of the kalman filter
SYST2  - system description file name for the original system
NAME2  - section name of the original system, SYST2
EVAL   - name of locus file containing desired poles
EPS    - test quantity
         by default EPS=the reserved varible REPS.
```

Function

The input system description matrices are read together with the derived reconstructor poles. The reconstructor filter gain is computed optionally together with the matrices of the reconstructor itself. Note that RECON will use the measured output Y. The reserved global variable REPS. is used in the transformation to observable canonical form.

Method

The given system (the equations for the discrete time case are analoguous)

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$$

is transformed to observable canonical form through $\xi = Tx$

$$\xi = TAT^{-1}\xi + TBu = A_\xi \xi + B_\xi u$$

$$y = CT^{-1}\xi + Du = C_\xi \xi + Du$$

The form of the matrices A_ξ and C_ξ is such that the filter gain

$$K = T^{-1} K_\xi$$

is immediately obvious, see the reference, page 252.

The optionally output reconstructor matrices are

$$\dot{x} = (A-KC)\ x + Bu + K \cdot y$$
$$y = I \cdot x$$

A-KC is the A matrix and K is the B_w matrix.

References

K.J. Astrom: Reglerteori, Almqvist & Wiksell, 1976.

Hints

The locus file EVAL may be given from POLES - PLEV as in the example below.

Examples

As example, we use the one-dimensional particle with the position as the output y. This is the same system SO as used in the example for the command LUEN.

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

with

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{and} \quad C = [\ 1 \quad 0\ ]$$

We now want to place both reconstructor poles in -1:

```
>POLES  EV < 50
>PLEV  ED < EV
   >ALT  1  -1.
   >ALT  2  -1.
   >X
```

Design
RECON

```
>SYST (SC) SOBS
   >INS BW
   >X
>RECON  K  SOBS < SO  ED
>LIST  K
```

The following numeric results are obtained:

$$K = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$SOBS:A = \begin{bmatrix} -2 & 1 \\ -1 & 0 \end{bmatrix} \qquad SOBS:B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$SOBS:BW = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \qquad SOBS:C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Referring to the original system, we find:

$$\dot{\hat{x}}_1 = -2\hat{x}_1 + \hat{x}_2 + 2y = \hat{x}_2 + 2(y - \hat{x}_1)$$

$$\dot{\hat{x}}_2 = -\hat{x}_1 + u + y = u + (y - \hat{x}_1)$$

Note that the reconstructor state still have the physical significance of the corresponding system state, cf. the command LUEN.

REDFB

Purpose

To compute an output feedback from a given state feedback. Both continuous time and discrete time cases are handled.

Command

REDFB K < SYST[(NAME)] L FBS W

K - approximate feedback matrix
SYST - system file name
NAME - name of section within SYST
L - state feedback matrix
FBS - feedback structure matrix, FBS(i,j) = 1 if connection between U(i) and Y(j) is permitted, else 0, for i = 1, 2, .., NU and j = 1, 2, .., NY
W - mode weighting matrix

Function

The input to the command is a system description on state space form and a state feedback matrix L giving a satisfactory closed loop performance. The result is an output feedback matrix K designed to give closed loop poles close to the positions obtained using L (if possible). Additional control of the result is available using the feedback structure matrix FBS and the weighting matrix W.

FBS specifies the elements of K that may have non-zero values: if FBS(i,j) = 0 then K(i,j) = 0, while if FBS(i,j) = 1 then K(i,j) ≠ 0 (maybe).

W is a diagonal matrix. The diagonal elements express the relative importance of keeping the corresponding eigenvalue unchanged. Eigenvalues (with feedback through L) are sorted in decreasing real parts.

REDFB uses the control input U and the measured output Y.

REDFB gives a plot of the eigenvalues. The eigenvalues of the original state feedback solution are marked with x, those of the output feedback solution are marked with o.

Method

The eigenvalues of the system (here we use continuous time formulations)

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

will, if closed with u = -Lx be those of

$$A - BL \qquad (1)$$

However, if it is closed with output feedback, i.e. u = -Ky, they are those of

$$A - BKC \qquad (2)$$

The problem now is to find a K such that the eigenvalues of (2) satisfactorily coincide with those of (1). Equality is generally not possible to achieve.

The problem is solved in REDFB in the manner proposed in the reference. Specifically, it is done by seeking the solution K^* minimizing the norm

$$\|(KCQ - LQ)W\| \qquad (3)$$

giving

$$K^* = LQW(CQW)^{\dagger} \qquad (4)$$

where $M^{\dagger}$ denotes the pseudoinverse of M.

$$W = \mathrm{diag}(W_1, W_2, \ldots)$$

is the mode weighting matrix given in the argument list, while Q is a matrix spanning the eigenspace of (A - BL), i.e. it contains the eigenvectors as columns

$$Q = [V_1, V_2, \ldots]$$

Using a suitable matrix norm (3) may be written as

$$\|(KCQ - LQ)W\| = \Sigma\, W_i^2\, |KC\, v_i - L\, v_i|^2 \qquad (5)$$

This indicates that the K^* of (4), minimizing (5), will be such that the eigenvalues and eigenvectors of (1) and (2) will be close. The effect of W is also seen! it determines which of the terms in (5) is the most critical and hence which eigenvalues that will change the least.

Reference

G. Bengtsson: A Theory for Control of Linear Multivariable Systems. Thesis TFRT-1006, Dept. of Automatic Control, Lund Institute of Technology, Sweden, 1973.

Cautions, Restrictions

REDFB does not guarantee a stable solution.

Hints

a) REDFB solves the problem of making a state feedback solution feasible when not all state variables are available for feedback and when a dynamic state reconstructor is not desired.

b) REDFB may also solve the problem of using a dynamic output compensator to partially reconstruct unavailable state variables, see the example.

c) REDFB may have to be used iteratively with different Ws to find an acceptable solution. Note that there may be cases where a stable solution does not exist, cf. the example.

Example

In the example used for the command OPTFB, the following system was used:

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

with

$$A = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

A state feedback L was found:

$$L = [\ 4.73347 \quad 13.5696 \quad 16.1716 \quad 8.51044\ ]$$

Now only two measurements were actually available, viz. state variables x_2 and x_4 as is seen in the C-matrix. A number of possibilities are now open, e.g. a full order state reconstruction, i.e. a Kalman tiler with the aid of KALFI, or a reduced order reconstruction as designed in LUEN.

A third possibility will be explored here. We will try to design an output feedback with REDFB. We assume the system BAB with feedback L to be available from the example in OPTFB.

```
>ENTER FBS  1  2             "Feedback structure
   #> 1  1                   "Use both outputs for a
                             "single input
   #> X
>UNITM  W  4                 "Initially equal weighting
>REDFB K < BAB L FBS W
>HCOPY                       "Copy to Figure 1
```

Figure 1 shows the displayed information from the call to REDFB. Obviously, the eigenvalues for the system closed with output feedback falls in the right half plane. Changing the value of W does not help in this case. An analysis of the system indicates that a proportional feedback from only the two measured outputs should not work. It does not give enough phase advance.

To introduce some phase advance, a dynamic output compensator with transfer function

$$G(s) = \frac{10\ s}{s + 10}$$

is connected to each output. G will produce an approximate derivative of the output for frequencies below 10 rad/s. The bandwidth is chosen not to interfere with the eigenvalues of the state feedback system. The state feedback L is expanded to be compatible with the new augmented system.

```
>SYST D < ABCD
>UNITM * -10  A  2
>UNITM * 10   B  2
>UNITM * -10  C  2
>UNITM * 10   D  2
>AGR D
   >INS A
   >INS B
   >INS C
   >INS D
   >X
>SYSOP SE < BAB D
   >IN U1 < U
```

```
    >IN U2 < Y1
    >OUT Y(1 2) < Y1
    >OUT Y(3 4) < Y2
    >X
>ZEROM LE  1  6
>EXPAN LE < LE  L(1 1)
>ENTER FBS  1  4
    #> 1  1  1  1
    #> X
>UNITM W  6
>REDFB K < SE  LE  FBS  W
```

The eigenvalues of the expanded system with feedback through LE are

$$\lambda = \{-1.05 \pm j\ 1.14,\quad -1.82 \pm j\ 0.48,\quad -10.,\quad -10.\}$$

The following gives an account of the system eigenvalues with the output feedback solution for various choices of W.

I. $W = \mathrm{diag}(1, 1, 1, 1, 1, 1)$
$\lambda = \{0.58 \pm j\ 0.77,\quad -1.09 \pm j\ 0.74,\quad -10,\quad -10\}$

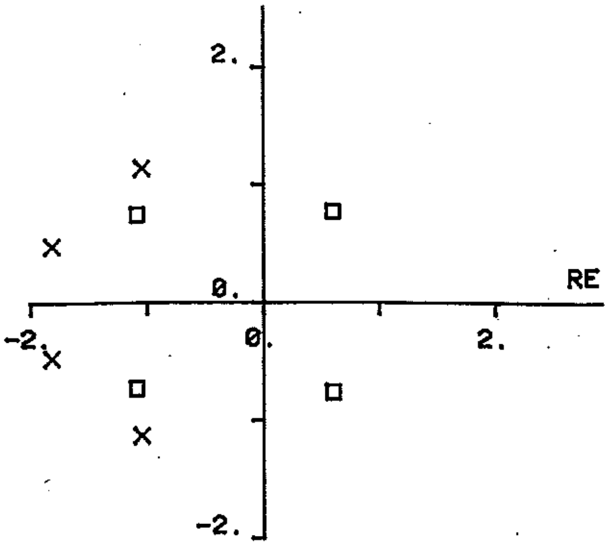
II. $W = \mathrm{diag}(1, 1, 1, 1, 0.01, 0.01)$
$\lambda = \{0.59,\quad -1.08 \pm j\ 1.1,\quad -1.79,\quad -7.6,\quad -10\}$

III. $W = \mathrm{diag}(10, 10, 10, 10, 0.01, 0.01)$
$\lambda = \{-1.04 \pm j\ 1.13,\quad -1.6 \pm 0.35,\quad -5.7,\quad -10\}$

This solution is selected as the final one. The value of K is

$$K = [7.9691 \quad 3.5785 \quad 2.3850 \quad 7.7245]$$

Compare this with the solution obtained with $\dot{y} = CAx$.



DELET

Purpose

Deletes files from the data base.

Command

```
DELET [(DMODE1)]FNAM1 [ [[(DMODE2)]FNAM2] ... ]

DMODE   - data mode indicator = 'D'/'T'/'A'
          D     - FNAM is assumed to contain binary data
          T     - FNAM is assumed to contain text
          A     - FNAM is assumed to be an aggregate file
(DMODE='D' by default)
FNAM    - file name
```

Function

If the specified file exists in the data base it is deleted; otherwise an error message is given.

Hints

The existence of the file may be tested by the command FTEST.

Example

Cf. the macro in the example in RANPA.

FHEAD

Purpose

To display file head parameters of data files and enable the user to change them.

Command

```
FHEAD [AGGREG:]FILE

AGGREG  - aggregate file name
FILE    - file name
```

Subcommands

```
INDEX VALUE
          set the INDEX:th parameter to VALUE
LOOK[K] - display the K:th (default all) file head
          parameter(s)
KILL    - exit from FHEAD without updating the file head
X       - exit from FHEAD and update the file head

Note    - an immediate, READ-ONLY form also exists:
          >FHEAD [AGGREG:]FILE 'LOOK' [K]  equivalent to:
          >FHEAD [AGGREG:]FILE
               >LOOK [K]
               >KILL
```

Function

The command has two different forms. One takes subcommands and also allows changes to be made. The other form is a read only form that does not enter subcommand mode. The output on the terminal is shown in the example below.

Cautions, Restrictions

It is not possible to change parameter 7. Attempts to alter parameters 1, 2, 3, and 10 will produce a warning message and should generally be avoided by users not very familiar with the internal orginazation of data files.

Hints

Reorganization of data files is often best done by special purpose commands like CUT, CONC, PICK, MOVE etc. In extreme cases the sequence FORMAT - (EDIT) - CONV could be used.

Examples

Two forms of the command are shown below with the accompanying output.

```
>FHEAD A
     >LOOK

FHEAD A

 1. NUMBER OF ROWS               2
 2. NUMBER OF COLUMNS            2
 3. THIRD DIMENSION              1
 4. SAMPLE INTERVAL              3 TICKS = 3.00000     S
 5. DATE RECORDER                0 (YY.MM.DD)
 6. TIME RECORDED                0 (HH:MM:SS)
 7. CONSTANT RECORD LENGTH 0     1 (1 MEANS YES)
 8. GENERATED BY COMMAND NR      2
 9. FILE TYPE                    3 = MATRIX
10. SKIP COUNT                   0

     >KILL

>FHEAD A LOOK 4

 4. SAMPLE INTERVAL              3 TICKS = 3.00000     S
```

FTEST

Purpose

To test the existence of files.

Command

```
FTEST [(DMODE)] FNAME

DMODE    - data mode indicator = 'A'/'D'/'T'
           A      - aggregate file
           D      - binary data file
           T      - text file
           (by default DMODE = 'D')
FNAME    - file name
FTEST.   - reserved variable returned = 0/1
           0      - the file does not exist
           1      - the file exists
```

Function

The command tries to access a file with the given name and type. If it was possible, the reserved global variable FTEST. is given the value 1, otherwise it is given the value 0.

TURN

Purpose

To alter the value of some program switches.

Command

TURN SWITCH STATE

```
SWITCH   - switch name = 'TEXT'/'TIME'/'GRAPH'/'DK'
           TEXT  - enables/disables all text output on
                   the display (default: TEXT is enabled)
           TIME  - if disabled, data will be plotted versus
                   sample number, else versus time units
                   (default: TIME is disabled)
           GRAPH - enables/disables all graphics output
                   (default: GRAPH is enabled)
           DK    - enables/disables command logging into
                   text files (default: DK is enabled)
STATE    - switch state = 'ON'/'OFF', if switch = 'TIME',
           'H'/'M'/'S' is used instead of 'ON'
           ON    - the switch is enabled
           H     - plotting will be versus time in hours
           M     - plotting will be versus time in minutes
           S     - plotting will be versus time in seconds
           OFF   - the switch is disabled
```

Function

The switches are set according to the value in the argument list.

Cautions, Restrictions

Note that the operation of the program also is controlled through the more flexible method of global variables, cf. the general guide.