3047

DISCO – A MICROCOMPUTER CONTROLLER FOR SINGLE–INPUT–SINGLE–OUTPUT SYSTEMS

L. ANDERSSON

Report 7601 July 1976
Department of Automatic Control
Lund Institute of Technology

DISCO - A MICROCOMPUTER CONTROLLER FOR SINGLE-INPUT-SINGLE-OUTPUT SYSTEMS

Leif Andersson

Department of Automatic Control
Lund Institute of Technology
Lund, Sweden

Abstract

The controller described in this report is mainly intended for education in control engineering. It is implemented on an Intel 8008 microcomputer, and admits the demonstration of various types of controllers such as PI, PID, minimum variance etc. The report contains a derivation of a suitable structure and a thorough description of the programs including full program lists.

Table of contents Page

1. INTRODUCTION

The importance of digital computers in control engineering is steadily increasing due to the decrease in hardware prices. The advent of the microprocessors has further enhanced this trend. In order to give students of control engineering some practical experience with both sampled data control system and microcomputer implementations the program DISCO, DIScrete time COntroller, and its associated hardware has been developed. DISCO is a controller for single input single output systems implemented on an Intel 8008 microprocessor. The controller parameters are set using a simple operator's panel with thumbwheel switches and a numerical display.

The system admits controllers that are dynamical systems up to order six. It is thus possible to demonstrate PI, PID, minimum variance, dead-beat controllers etc. In the laboratory experiments the plant dynamics is simulated on analog computer. DISCO is then connected to its input and output and the closed loop performance is investigated.

The report contains a short description of the hardware and a derivation of a suitable controller structure. In order to give an overview of the programs, their actions are described in PASCAL, a high level programming language. The performance of DISCO in terms of memory requirement and execution times are described together with some test examples.

The appendices contain full program listings and, for reference purposes, a summary of state observer theory.

2. THE HARDWARE

DISCO is implemented on an Intel 8008 microprocessor. The CPU, associated circuitry and memory is a commercially available unit. A process interface containing one D/A and one A/D converter and a real time clock was designed, as well as a simple operator's console used for entering control parameters. Fig 1 gives an overall picture of the equipment.

2.1 The CPU

A brief description of the Intel 8008 is given here for reference purposes.

The Intel 8008 is an eight bit CPU containing one accumulator and seven 8-bit data registers. All arithmetics is performed between the accumulator and the data registers or the memory. The instruction time ranges between 20 μs for simple register to register operation, and 44 μs for jump instructions. It has one interrupt level, but on this level eight different interrupt sources may be directly recognized. The address space is 16 K memory, which may be mixed RAM and ROM in any combination.

2.2 The Process Interface

The A/D converter is a ZELTEX ZD460 8-bit converter. The input voltage is in the range -10 V - +10 V, the conversion time is 50 μs and the output is a two's complement straight binary number.

The D/A converter is an 8-bit ZELTEX ZD430 accepting a straight binary two's complement number and giving -10 V - +10 V. The conversion time is 25 μs.

The real time clock is based on an astable multivibrator with a period of 0.125 sec. This frequency is divided by a front panel selectable factor of 1, 2, 4, 8 or 16, and then applied to the interrupt line. The front panel of the interface also contains a manual interrupt pushbutton, a clock on/off switch, external interrupt input, external clock on/off input and clock output. See fig 2.

2.3 The Operators Console

Fig 3 shows the console front panel. It contains two sets of thumbwheel switches, one for address input (1) and the other for entering data (2). The contents of the selected memory call is showed on a $3\frac{1}{2}$ digit LED (3). Three switches (4) determine the interpretation of the cell contents before display. The position OCTAL-UNSIGNED gives the content as an octal number in the range 0-377. OCTAL-SIGNED gives an octal number -200 to +177, DECIMAL-INTEGER gives a decimal number -128 to +127 and DECIMAL-FRACTION gives a fractional number -1.000 to +0.992.

The number on the data switches (2) is entered into memory when the IN-button (5) is pressed. The interpretation of the number is the same as for the display.

The program necessary to drive this operators console occupies approximately 0.5 K of memory. It is listed in appendix B.

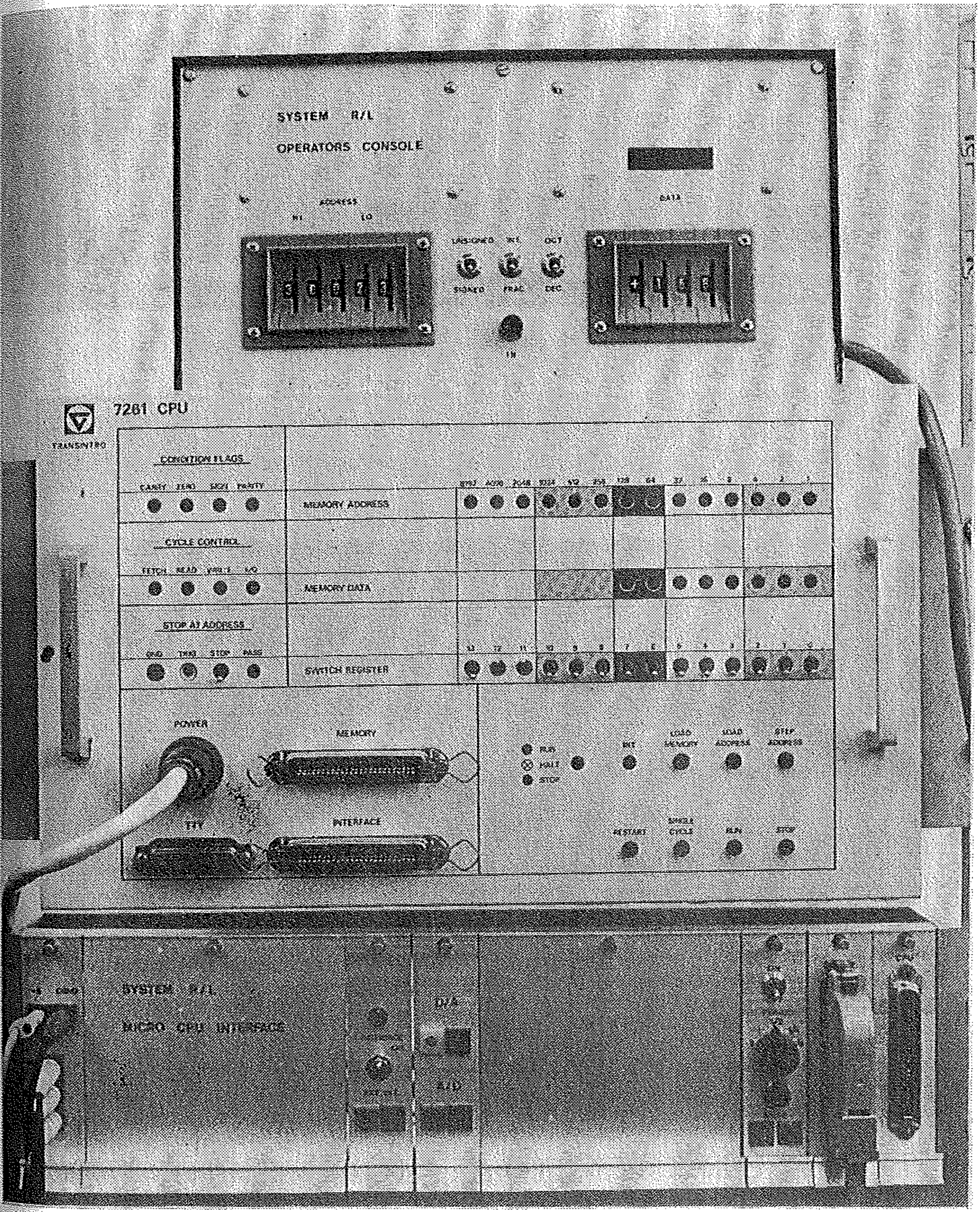


Fig 1. CPU, Interface and Operator's Console.

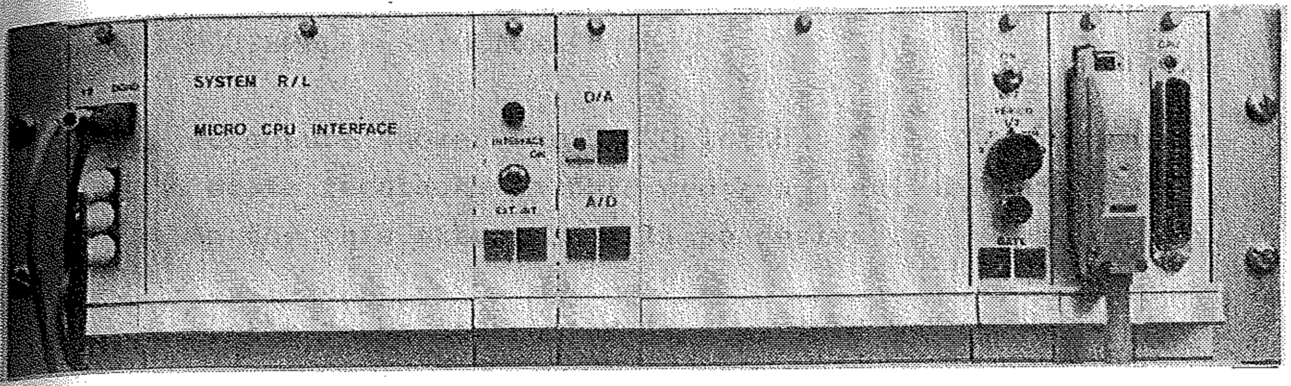


Fig 2. Interface front panel.

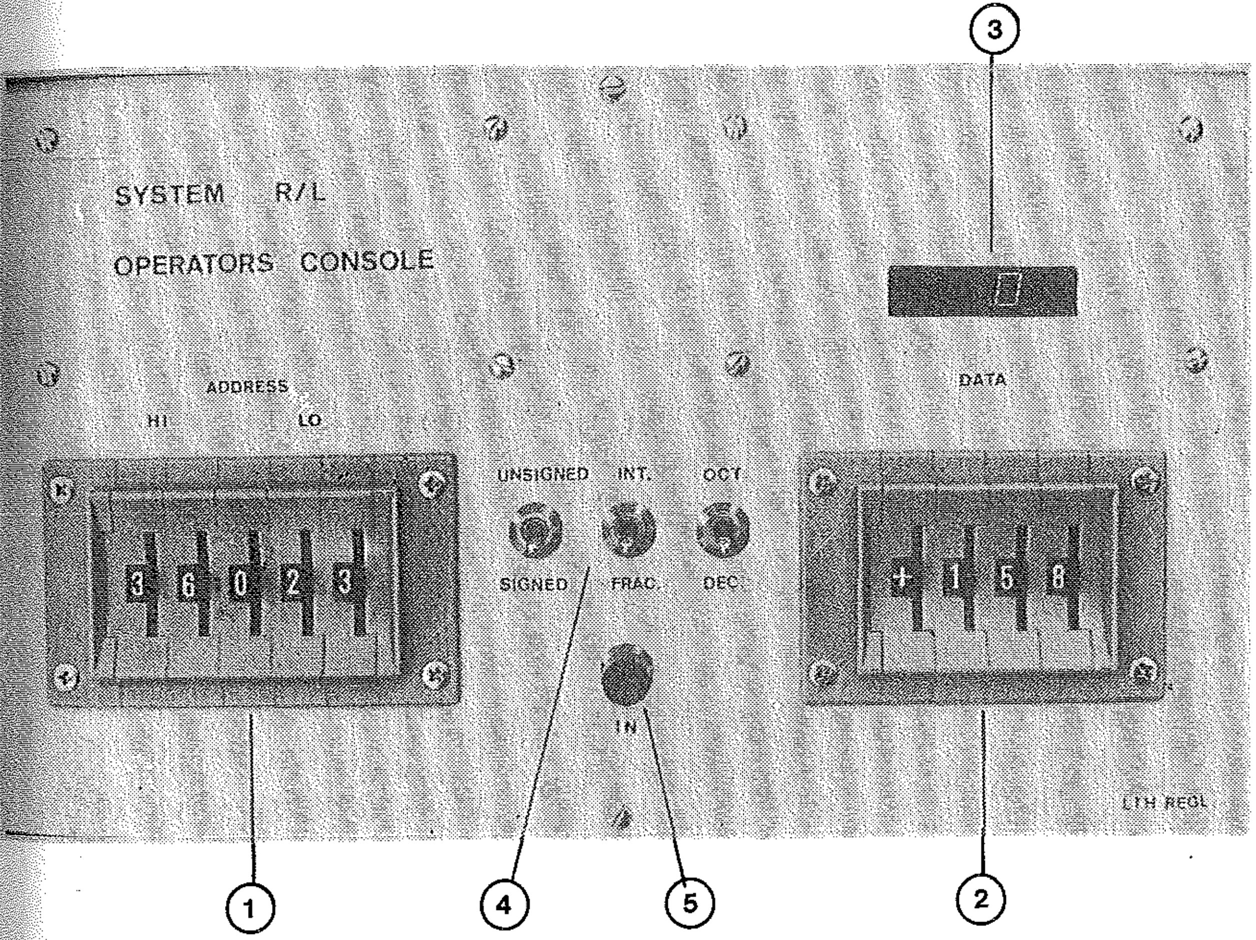


Fig. 3. Operator's Console.

3. CONTROLLER STRUCTURE

The structure chosen for DISCO is that of state estimation plus state feedback. In this section a suitable implementation for this structure will be derived.

The system to be controlled is described by

$$y(t) = \frac{B(q)}{A(q)} u(t) \tag{3.1}$$

where q is the forward shift operator, $qy(t) = y(t+1)$, and A and B are polynomials

$$\begin{aligned} A(q) &= q^n + a_1 q^{n-1} + \ldots + a_n \\ B(q) &= b_1 q^{n-1} + \ldots b_n \end{aligned} \tag{3.2}$$

The polynomials A and B are assumed to be relatively prime.

Introduce a suitable state space representation:

$$\begin{aligned} x(t+1) &= \phi\ x(t) + \Gamma\ u(t) \\ y(t) &= [\,1 \quad 0 \ldots 0\,]\ x(t) \end{aligned} \tag{3.3}$$

and partition the state vector into x_1 of length 1 and x_r of length n-1:

$$x(t) = \begin{pmatrix} x_1(t) \\ x_r(t) \end{pmatrix} = \begin{pmatrix} y(t) \\ x_r(t) \end{pmatrix}$$

The theory for reduced order state estimators [1] states that it is possible to find a dynamical system of order n-1:

$$\begin{aligned} \hat{z}(t+1) &= \phi_r\ \hat{z}(t) + \Gamma_{r1}\ u(t) + \Gamma_{r2}\ y(t) \\ \hat{x}_r(t) &= \hat{z}(t) + K\ y(t) \end{aligned} \tag{3.4}$$

where the eigenvalues of ϕ_r may be arbitrarily specified, and such that $\hat{x}_r$ is a reconstruction of x_r with the reconstruction error $\tilde{x}_r = x_r - \hat{x}_r$ obeying

$$\tilde{x}_r(t+1) = \phi_r \tilde{x}_r(t) \tag{3.5}$$

A summary of the theory will be found in appendix C.

Introduce a reference input $y_r(t)$ and a state feedback

$$u(t) = y_r(t) - L\hat{x}(t) = y_r(t) - [\ell_1 \; L_r] \begin{bmatrix} y(t) \\ \hat{x}_r(t) \end{bmatrix} = y_r(t) - \ell_1 y(t) - L_r \hat{x}_r(t) \tag{3.6}$$

such that the matrix $\phi - \Gamma L$ gets its eigenvalues in desired positions. The controller is then a dynamical system of order n-1:

$$\begin{aligned} \hat{x}_r(t+1) &= [\phi_r - \Gamma_{r1} L_r] \hat{x}_r(t) + [\Gamma_{r2} - \Gamma_{r1}\ell_1] y(t) + \Gamma_{r1} y_r(t) \\ u(t) &= y_r - L_r x_r(t) - \ell_1 y(t) \end{aligned} \tag{3.7}$$

The closed loop system becomes

$$\begin{pmatrix} x(t+1) \\ \tilde{x}_r(t+1) \end{pmatrix} = \begin{pmatrix} \phi - \Gamma L & \Gamma L_r \\ 0 & \phi_r \end{pmatrix} \begin{pmatrix} x(t) \\ \tilde{x}_r(t) \end{pmatrix} + \begin{pmatrix} \Gamma \\ 0 \end{pmatrix} y_r(t)$$

$$y(t) = [1 \;\; 0 \;\; 0 \ldots 0] \; x(t) \tag{3.8}$$

This system is of order $n + n - 1$ and the n-1 poles belonging to the estimation error are uncontrollable.

The controller (3.7) may be rewritten in pulse transfer function form:

$$u(t) = \begin{pmatrix} \dfrac{-R(q)}{S(q)} & \dfrac{P(q)}{S(q)} \end{pmatrix} \begin{pmatrix} y(t) \\ y_r(t) \end{pmatrix} \tag{3.9}$$

where P, R and S are polynomials of order n-1:

$$P(q) = p_0\, q^{n-1} + p_1\, q^{n-2} + \ldots + p_{n-1}$$

$$R(q) = r_0\, q^{n-1} + \ldots\ldots\ldots\ldots + r_{n-1}$$

$$S(q) = q^{n-1} + s_1\, q^{n-2} + \ldots\ldots + s_{n-1}$$

With the controller (3.9) the closed loop system is

$$y(t) = \frac{B(q)P(q)}{A(q)S(q) + B(q)R(q)} \qquad (3.10)$$

The coefficients of P, R and S may be determined directly as follows:
Let the deisred characteristic polynomial be

$$D(q) = q^{2n-1} + d_1\, q^{2n-2} + \ldots + d_{2n-1}$$

Multiplying A and S, B and R gives the following equation system

$$\begin{pmatrix} 1 & & 0 & b_1 & & 0 \\ a_1 & \ddots & & \vdots & \ddots & \\ \vdots & \ddots & 1 & \vdots & & b_1 \\ a_n & & a_1 & b_n & & \vdots \\ & \ddots & \vdots & & \ddots & \vdots \\ 0 & & a_n & 0 & & b_n \end{pmatrix} \begin{pmatrix} s_1 \\ \vdots \\ s_{n-1} \\ r_0 \\ \vdots \\ r_{n-1} \end{pmatrix} = \begin{pmatrix} d_1 - a_1 \\ \vdots \\ d_n - a_n \\ d_{n+1} \\ \vdots \\ d_{2n-1} \end{pmatrix} \qquad (3.11)$$

This equation has a unique solution if and only if A and B are relatively prime [2].

Equation (3.8) states that the closed loop system should have n-1 uncontrollable poles. This gives a condition on the polynomial P(q): it should be chosen so that it cancels n-1 of the closed loop poles.

The steady state gain of the closed loop system is given by

$$\frac{y(\infty)}{y_r(\infty)} = \frac{\sum_{1}^{n} b_i \cdot \sum_{0}^{n-1} p_i}{\sum_{0}^{n} a_i \cdot \sum_{0}^{n-1} s_i + \sum_{1}^{n} b_i \cdot \sum_{0}^{n-1} r_1} \tag{3.12}$$

where $a_0 = s_0 = 1$.

This gives a final condition for P. The coefficients should be chosen so that the steady state gain has a desired value.

If it is desirable to have an integrating controller, i.e. if S(q) should contain a factor q-1, then the design problem is solved as follows:

Introduce the polynomials $\overline{S}(q)$ and $\overline{A}(q)$ such that

$$\begin{aligned} S(q) &= \overline{S}(q)\ (q-1) \\ \overline{A}(q) &= A(q)\ (q-1) \end{aligned} \tag{3.13}$$

In order to completely determine the closed loop poles in this case R(q) must be of order n. The coefficients of $\overline{S}$ and R are then given by an equation system like (3.11) but with n-1 columns containing the coefficients of $\overline{A}$ and n+1 columns containing b_i. Note that in this case the B-polynomial must not have a zero in q = 1.

If either the process or the controller contains an integrator, the expression (3.12) for the steady state gain is reduced to

$$\frac{y(\infty)}{y_r(\infty)} = \frac{\Sigma p_i}{\Sigma r_i} \tag{3.14}$$

The controller (3.9) is expressed in the forward shift operator q. In order to implement it, it must naturally be expressed in the backward shift operator q^{-1}:

$$S^*(q^{-1})\ u(t) = P^*(q^{-1})\ y_r(t) - R^*(q^{-1})\ y(t) \qquad (3.15)$$

where $P^*(q^{-1}) = p_0 + p_1 q^{-1} + \ldots + p_{n-1} q^{n-1}$ etc.

In explicit form we get

$$\begin{aligned} u(t) = {} & p_0 y_r(t) + p_1 y_r(t-1) + \ldots + p_{n-1} y_r(t-n+1) - \\ & - r_0 y(t) - r_1 y(t-1) - \ldots - r_{n-1} y(t-n+1) - \qquad (3.16) \\ & - s_1 u(t-1) - s_2 u(t-2) - \ldots - s_{n-1} u(t-n+1) \end{aligned}$$

which may be implemented as a sum of scalar products of vectors containing the coefficients of P, R, S and old values of y_r, y and u. See also fig 4.

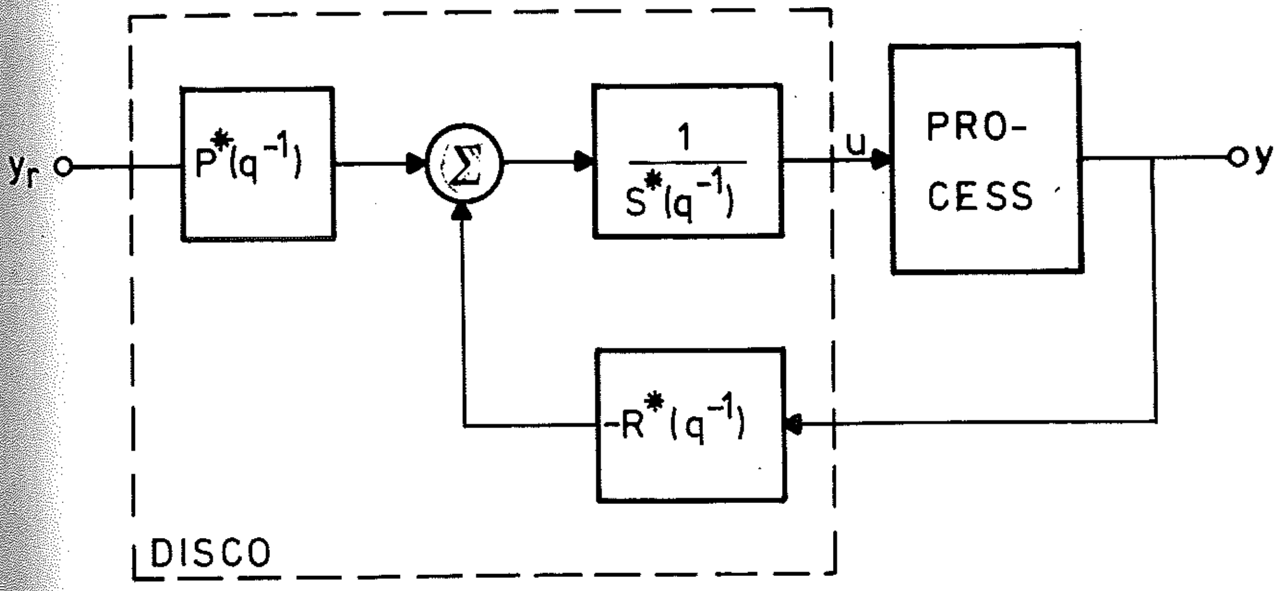


Fig 4. Regulator structure.

4. PROGRAM IMPLEMENTATION

The programming of DISCO was done in three levels. First the data base and the main program was described in the programming language PASCAL [3]. Naturally no compiler exists which can translate PASCAL into Intel 8008 machine code, but in the author's opinion writing in a high level language is a good alternative to flow diagrams. An additional advantage is that the data base may be formally described in PASCAL.

The PASCAL code was then hand-translated into MLP, a medium level language for the 8008, which was originally developed by C.E.R.L in England [4], and which has been modified [5] to include address variables and some features of PASCAL.

The output from the MLP compiler is assembly language which was then passed to the assembler and loader [6].

4.1 Data Representation

Vectors are represented in a special floating point form, with one exponent for the entire vector. The exponent is a two's complement eight-bit number, and the elements are two's complement, eight-bit fractional numbers. This method of representing vectors simplifies scalar products, compared to one exponent per element, without excessive loss in accuracy.

The test of whether overflow has occurred or not as a result of an addition is a rather complex operation on the 8008. To avoid this problem the result of a multiplication is always given as three eight-bit words. The two least significant words are of course the product itself, while the extra most significant word contains the sign of the result.

All three words then take part in the additions, and no overflow testing is necessary until the last step when the result is converted to a fixed point number.

The result of a scalar product is then a floating point number with a three word two's complement mantissa, where the decimal point is immediately to the right of the MSB of the second word. The exponent is an eight bit two's complement integer.

4.2 Program Flow

It is important that the time interval between the reading of the process values and the writing of the controller output is as short as possible. With the structure chosen it is possible to perform a large part of the computations in advance, i.e. during the previous sampling interval. As can be seen from the controller expression (3.16) it is only the terms $y_r(t)\cdot p_0$ and $y(t)\cdot r_0$ which cannot be computed in advance. The program flow is thus as follows:

When DISCO is triggered by the clock the term $y_r(t)\cdot p_0$ is computed and added to the previously computed and added to the previously computed value. Then the A/D converter is read and the term $y(t)\cdot r_0$ is computed and added. The controller output is now complete and it is written on the D/A converter. The vectors y_r, y and u are then moved one step upwards and the computations in preparation for the next sampling instant are performed. Control is then transfered to the operator's communication program OPCOM, which runs as a background program when DISCO is idle. Since OPCOM runs in an infinite loop reading the address switches and displaying the contents, it is not necessary to save the processor status on interrupt. OPCOM is simply restarted when DISCO is finished.

4.3 DISCO Expressed in PASCAL

Since the PASCAL source program will never be passed to a compiler, a few liberties has been taken with the semantics. The deviations from the rules of standard PASCAL are:

1. A function may have a value of record type.
2. A variable of record type may be assigned a value through a single assignment statement, in which case all the fields get that value.
3. The character used as comment delimiter seems to vary from implementation to implementation. Here the double quote (") is used.
4. The data type WORD is considered predeclared. It consists of one eight-bit number. It may represent different things as shown below.

Some basic routines such as fixed point multiplication and others are not suitable to express in PASCAL. In these cases the program body contains just a comment describing the action of the routine.

```
PROGRAM DISCO
        "DISCRETE TIME CONTROLLER FOR SINGLE INPUT - SINGLE OUTPUT
         SYSTEMS"

TYPE    SINGLE=WORD;
          "THE TYPE SINGLE REPRESENTS A FIXED POINT,TWO'S COMPLEMENT
           FRACTIONAL NUMBER WITH THE DECIMAL POINT IMMEDIATELY TO
           THE RIGHT OF THE SIGN BIT"
        TRIPLE=RECORD M1,M2,M3:WORD
          "TRIPLE REPRESENTS A FIXED POINT, TWO'S COMPLEMENT REAL
           NUMBER WITH THE DECIMAL POINT IMMEDIATELY TO THE RIGHT
           OF THE MSB OF M2"
        END;
        FLOAT=  RECORD  EXP:INTEGER;
                        MANT:TRIPLE
        END;
        VECTOR=RECORD   EXP:INTEGER;
                        V:ARRAY[0..7] OF SINGLE
        END;

VAR     YREF:SINGLE;    "REFERENCE VALUE, ADDRESS 0"
        PNUM:INTEGER;   "LENGTH OF P, ADDRESS 1"
        RNUM:INTEGER;   "LENGTH OF R, ADDRESS 2"
        SNUM:INTEGER;   "LENGTH OF S, ADDRESS 3"
        SWITCH:INTEGER; "IF SWITCH=0 THEN RUN THE REGULATOR ELSE
                         ZERO THE VECTORS YR,U,Y, THE TEMPORARIES
                         TP,TR,TS AND THE ANALOG OUTPUT."
        P:VECTOR;       "FEEDFORWARD POLYNOMIAL, ADDRESS 10"
        R:VECTOR;       "FEEDBACK POLYNOMIAL, ADDRESS 20"
        S:VECTOR;       "REGULATOR DENOMINATOR, ADDRESS 30"
        YR:VECTOR;      "REFERENCE VALUES, ADDRESS 40"
        Y:VECTOR;       "MEASURED VALUES, ADDRESS 50"
        U:VECTOR;       "CONTROL VALUES, ADDRESS 60"
        TP,TR,TS:FLOAT; "TEMPORARY STORAGE"

FUNCTION SCAPR(A,B:VECTOR; LENGTH: INTEGER): FLOAT;
        "COMPUTES SCALAR PRODUCT, IF LENGTH=0 THEN SCAPR:=0"
VAR     T: FLOAT; I: INTEGER;   "TEMPRARY STORAGE"
BEGIN   T:=0;
        IF LENGTH /=0 THEN BEGIN
          T.EXP:=A.EXP+B.EXP;
          FOR I:=0 TO LENGTH-1 DO
            T.MANT:=ADD3(MULT(A.V[I],B.V[I]),T.MANT);
        END;
        SCAPR:=T
END;

PROCEDURE VMOVE(REF A: VECTOR; LENGTH: INTEGER);
        "MOVES A VECTOR ONE STEP UPWARDS IN MEMORY AND ZEROES
         THE BOTTOM ELEMENT"
VAR     I: INTEGER;

BEGIN   IF LENGTH>0 THEN BEGIN
          FOR I:=LENGTH-1 DOWNTO 1 DO
            A.V[I]:=A.V[I-1];
        END;
        A.V[0]:=0
END;
```

```
FUNCTION ADD3(T1,T2: TRIPLE): TRIPLE;
BEGIN    "FIXED POINT ADDITION"  END;

FUNCTION SUBF(F1,F2: FLOAT): FLOAT;
BEGIN    "FLOATING POINT SUBTRACTION"  END;

FUNCTION MULT(S1,S2: SINGLE): TRIPLE;
BEGIN    "FIXED POINT MULTIPLICATION"  END;

FUNCTION ADIN: SINGLE;
BEGIN    "READS THE A/D CONVERTER"  END;

PROCEDURE DAOUT(S: SINGLE);
BEGIN    "WRITES THE VALUE OF S ON THE D/A CONVERTER "  END;

FUNCTION FIX(F: FLOAT): SINGLE;
BEGIN    "CONVERTS A FLOATING POINT NUMBER TO A FIXED POINT
          NUMBER, IF THE CONVERSION GIVES OVERFLOW
          THE RESULT IS SET PLUS OR MINUS FULL SCALE"
END;

PROCEDURE OPCOM;
BEGIN    "OPCOM IS THE OPERATOR'S COMMUNICATION ROUTINE
          THROUGH WHICH MOST OF THE VARIABLES GET
          THEIR VALUES"
END;

PROCEDURE INIT;
         "INITIALISATION PROCEDURE"
BEGIN    SWITCH:=0;
         YR:=0;
         Y:=0;
         U:=0;
         TP.MANT:=0;
         TR.MANT:=0;
         TS.MANT:=0;
         TP.EXP:=P.EXP;
         TR.EXP:=R.EXP;
         TS.EXP:=S.EXP;
         DAOUT(0)
END;
```

```
BEGIN   "MAIN PROGRAM"
        IF SWITCH/=0 THEN INIT ELSE
        BEGIN

          "COMPUTE THE LAST TERMS OF YR*P, SUBTRACT THE PREVIOUSLY
           COMPUTED U*S"

          YR.V[0]:=YREF;
          TP.MANT:=ADD3(MULT(YR.V[0],P.V[0]),TP.MANT);
          TP:=SUBF(TP,TS);

          "READ PROCESS VALUE,COMPUTE THE LAST TERM OF Y*R,
           SUBTRACT AND WRITE THE CONTROL SIGNAL."

          Y.V[0]:=ADIN;
          TR.MANT:=ADD3(MULT(Y.V[0],R.V[0]),TR.MANT);
          TP:=SUBF(TP,TR);
          U.V[0]:=FIX(TP);
          DAOUT(U.V[0]);

          "START COMPUTING FOR NEXT SAMPLE"

          VMOVE(YR,PNUM);
          VMOVE(Y,RNUM);
          TS:=SCAPR(U,S,SNUM);
          TP:=SCAPR(YR,P,PNUM);
          TR:=SCAPR(Y,R,RNUM);
          VMOVE(U,SNUM)
        END;
        OPCOM
END.
```

4.4 Memory Requirements, Execution Times etc.

The total memory requirement for DISCO is 536 words. Of these the main program occupies 150 words, the scalar product 131 and the fixed point multiplication 96 words. The driver program for the operators console takes 480 words. The total amount of program memory is thus approximately 1 k words.

The data requires some 60 words of RAM memory.

The execution time may be approximated by the linear formula

$$t = 10 + 4\,p$$

where t is the execution time in ms and p is the number of parameters.

The time between clock pulse and A/D conversion is 4.4 ms and between A/D and D/A 5.3 ms. The fixed point multiplication takes 2.4 ms.

5 : 1

5. EXPERIMENTS WITH DISCO

5.1 Test Examples

Two second order time-continuous systems has been chosen as test examples.

System 1 (minimum phase):

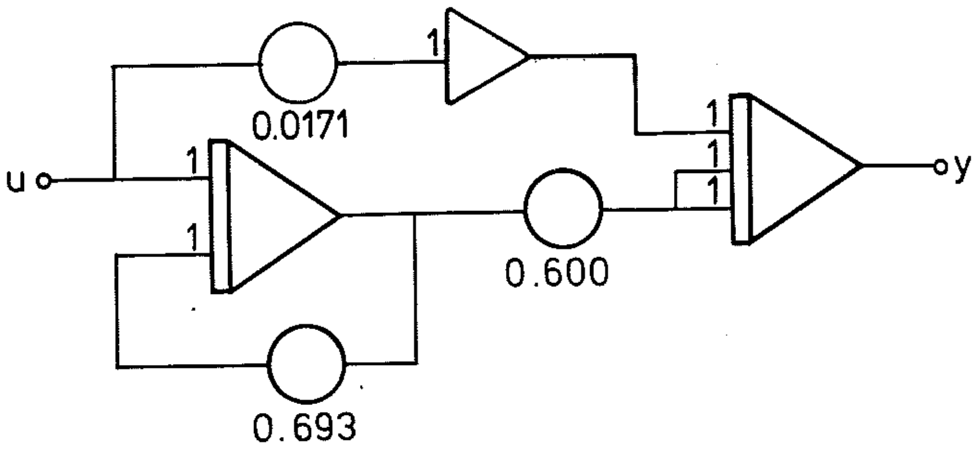
$$G(s) = \frac{1.213 + 0.0171\ s}{s(s + 0.693)}$$

A state space representation is

$$\dot{x} = \begin{pmatrix} -0.693 & 0 \\ 1.201 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0.0171 \end{pmatrix}$$

$$y = \begin{pmatrix} 0 & 1 \end{pmatrix} x$$

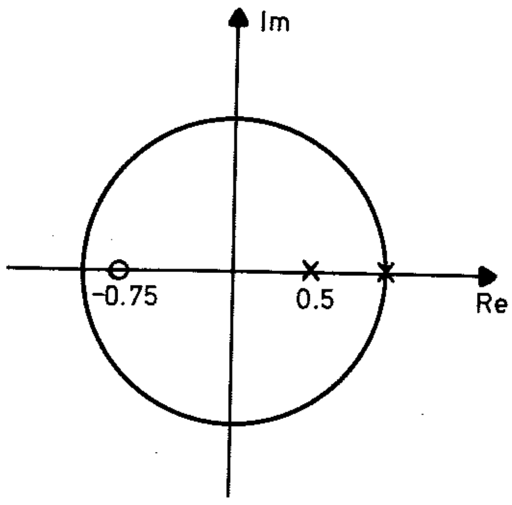
which gives the following diagram for analog computer simulation:



A sampled-data representation of this system with sampling period 1 sec. is

$$H^*(q^{-1}) = \frac{B^*(q^{-1})}{A^*(q^{-1})} = \frac{0.5\ q^{-1} + 0.375\ q^{-2}}{1 - 1.5\ q^{-1} + 0.5\ q^{-2}}$$

with the following pole-zero configuration:



System 2 (non-minimum phase):

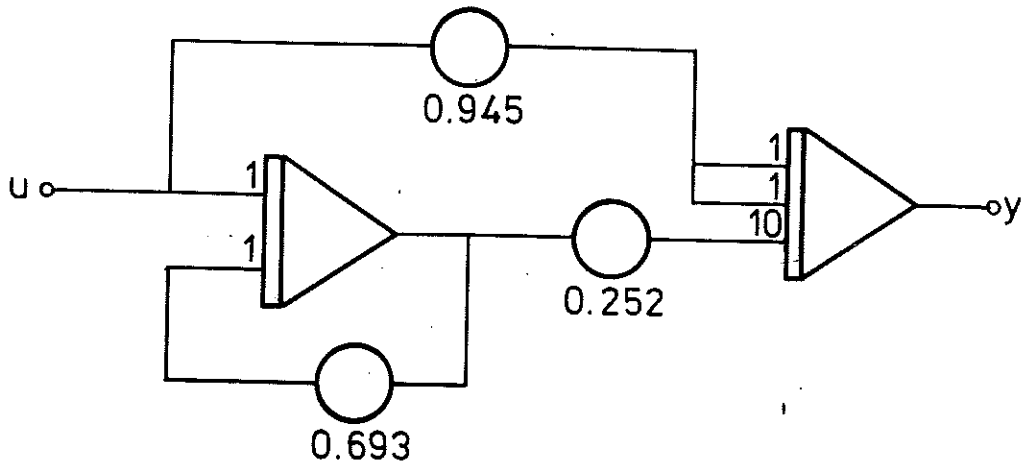
$$G(s) = \frac{1.213 - 1.889\,s}{s(s+0.693)}$$

State space representation:

$$\dot{x} = \begin{pmatrix} -0.693 & 0 \\ 2.522 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ -1.889 \end{pmatrix}$$

$$y = \begin{pmatrix} 0 & 1 \end{pmatrix} x$$

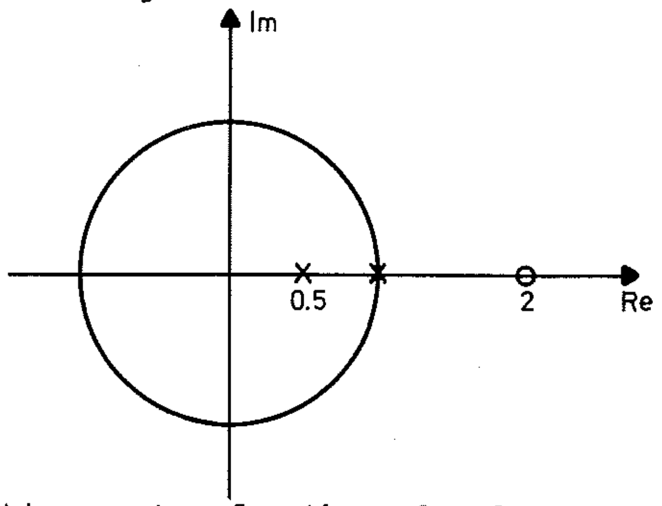
Analog computer diagram:



Sampled data representation:

$$H^*(q^{-1}) = \frac{-0.875\, q^{-1} + 1.75\, q^{-2}}{1 - 1.5\, q^{-1} + 0.5\, q^{-2}}$$

Pole-zero configuration:



The equation system for the pole placement problem for system 1 is

$$\begin{pmatrix} 1 & 0.5 & 0 \\ -1.5 & 0.375 & 0.5 \\ 0.5 & 0 & 0.375 \end{pmatrix} \begin{pmatrix} s_1 \\ r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} d_1 + 1.5 \\ d_2 - 0.5 \\ d_3 \end{pmatrix}$$

where d_i are the coefficients of the desired closed loop characteristic polynomial.

For system 2 the equation system is

$$\begin{pmatrix} 1 & -0.875 & 0 \\ -1.5 & 1.75 & -0.875 \\ 0.5 & 0 & 1.75 \end{pmatrix} \begin{pmatrix} s_1 \\ r_0 \\ r_1 \end{pmatrix} = \begin{pmatrix} d_1 + 1.5 \\ d_2 - 0.5 \\ d_3 \end{pmatrix}$$

An integrating regulator for system 1 is obtained as follows:

$$\bar{A}(q) = (q-1)A(q) = (q^2-1.5q+0.5)(q-1) = q^3 - 2.5q^2 + 2q - 0.5$$

The equation system is then

$$\begin{pmatrix} 1 & 0.5 & 0 & 0 \\ -2.5 & 0.375 & 0.5 & 0 \\ 2 & 0 & 0.375 & 0.5 \\ -0.5 & 0 & 0 & 0.375 \end{pmatrix} \begin{pmatrix} \bar{s}_0 \\ r_0 \\ r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} d_1 + 2.5 \\ d_2 - 2 \\ d_3 + 0.5 \\ d_4 \end{pmatrix}$$

and the S-polynomial becomes

$$S(q) = (q + \bar{s}_0)(q-1)$$

5.2 Criteria for the Pole Placement

For continuous time system the criteria for the pole placement are well known: the real part of the rightmost poles determine the resolution time, and the angle between the negative real axis and the line from the origin to the dominant poles determine the damping. It is less well known that similar criteria exist for discrete time systems. In this case the critical curves are obtained by mapping two sets of straight lines by the map e^{-sT_0} where T_0 is the sampling period. The first set is the line of constant real parts, which is mapped as circles. The second set is the locus of equal damping, i.e. straight lines from the origin. The image of this set consists of logarithmical spirals. See fig 5.
The parametrization of the spirals is damping values and of the circles the corresponding real part.

With this set of curves available it is quite simple to decide suitable positions for the closed loop poles.

1. Fix the damping for the dominant pole pair.

2. Determine the desired resolution time. This involves a compromise since the shorter the resolution time is,

the larger control signals are necessary. These two points give the position of the dominant poles.

3. Place the remaining poles on the real axis closer to the origin than the dominant poles. Again the closer to the origin the poles are placed, the larger the control signal becomes.

5.3 Test results

The following pages show the step responses for the test systems with various positions for the closed loop poles. In all the examples the observer pole has been placed in the origin.

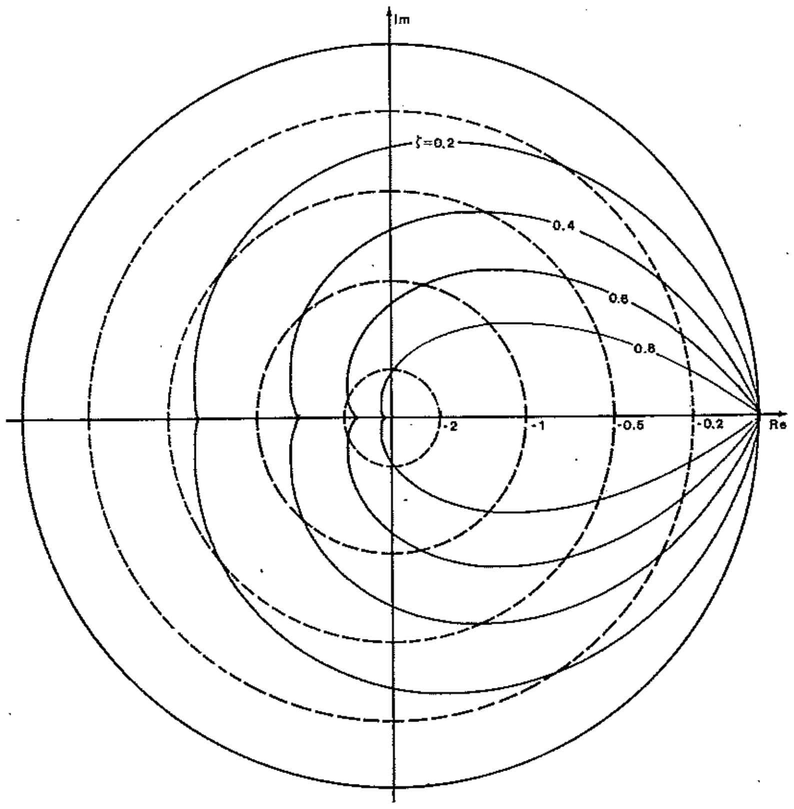
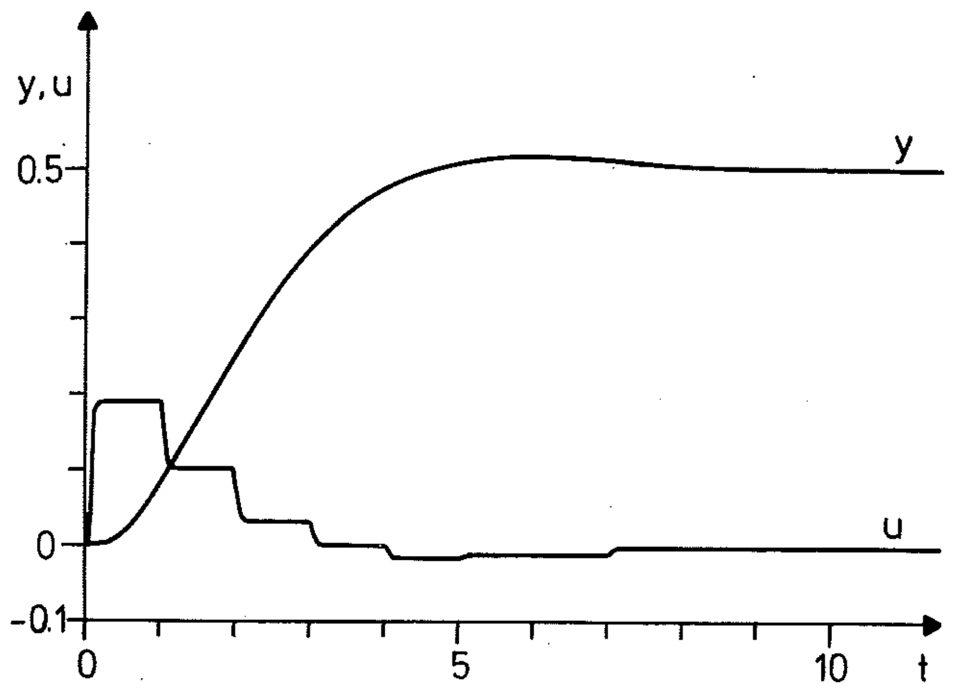


Fig 5. Images of constant damping and constant real part.



System 1. Damping 0.7. Circle -0.5.

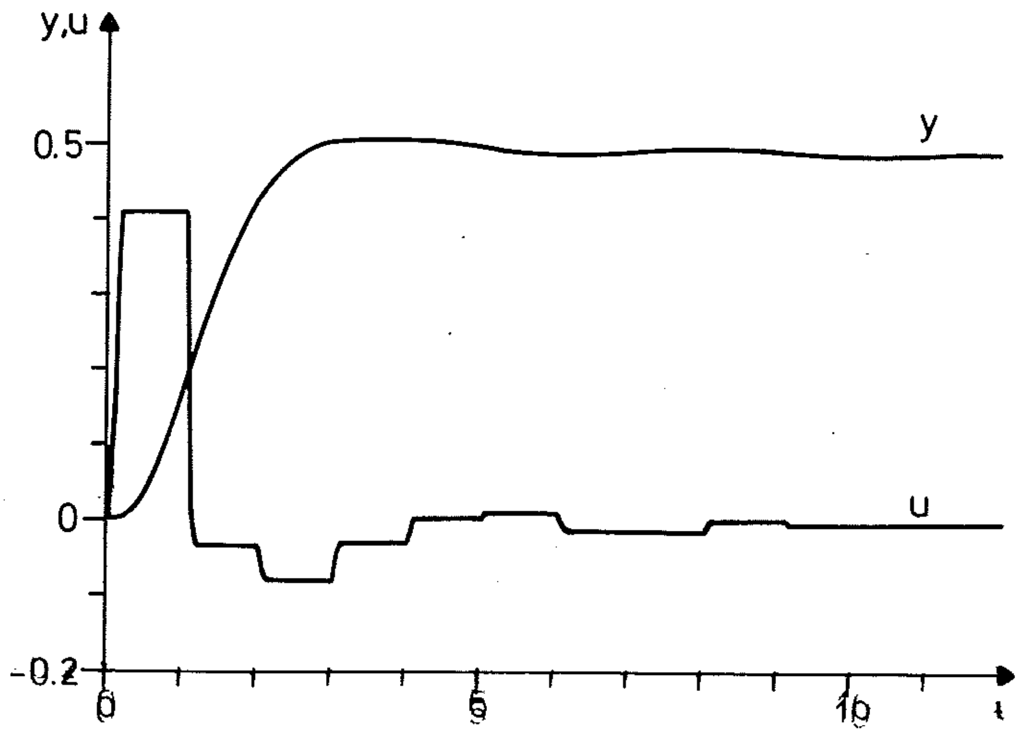
Poles in 0, $0.5 \pm i0.3$

Characteristic polynomial $q^3 - q^2 + 0.34q$

Regulator: $P^*(q^{-1}) = 0.388$

$R^*(q^{-1}) = 0.633 - 0.245q^{-1}$

$S^*(q^{-1}) = 1 - 0.183q^{-1}$



System 1. Damping 0.7. Circle -1.

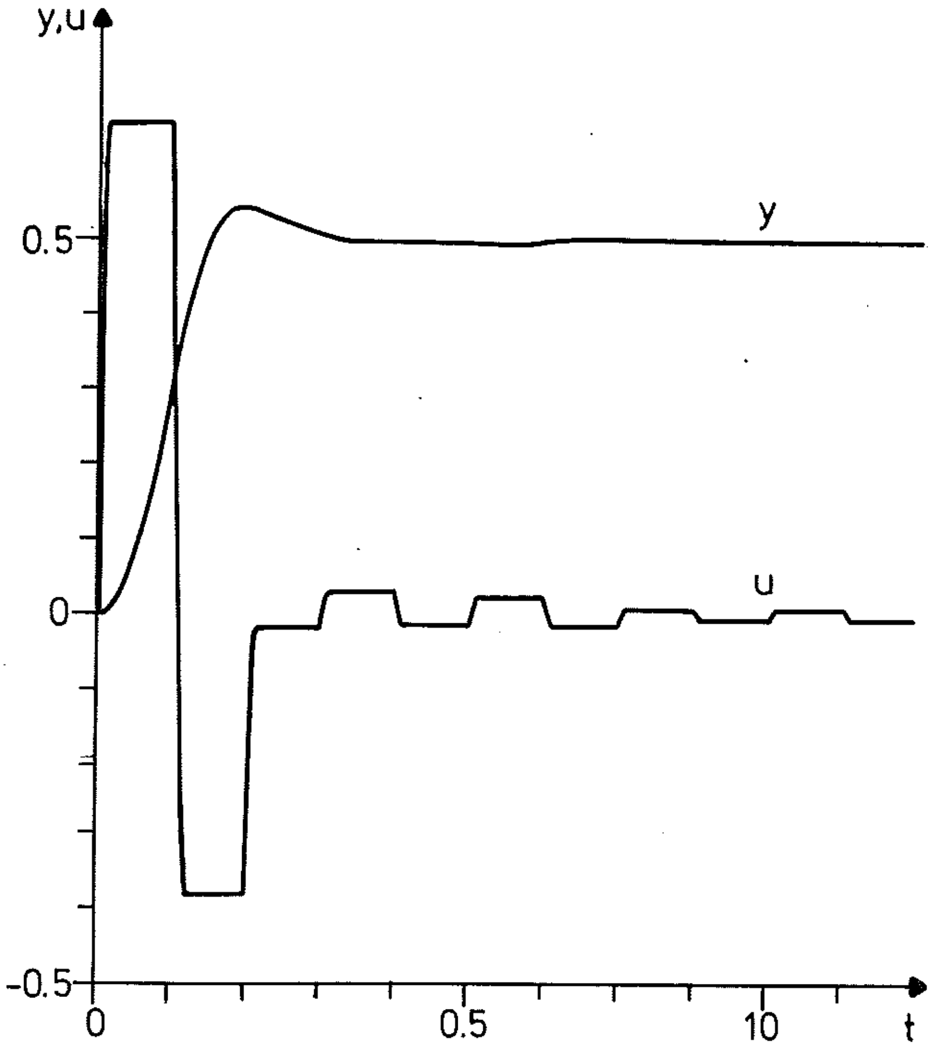
Poles in 0, $0.2 \pm i0.3$

Characteristic polynomial $q^3 - 0.4q^2 + 0.13q$

Regulator: $P^*(q^{-1}) = 0.835$

$R^*(q^{-1}) = 1.381 - 0.546q^{-1}$

$S^*(q^{-1}) = 1 + 0.410q^{-1}$



System 1. Damping 0.7. Circle -2.

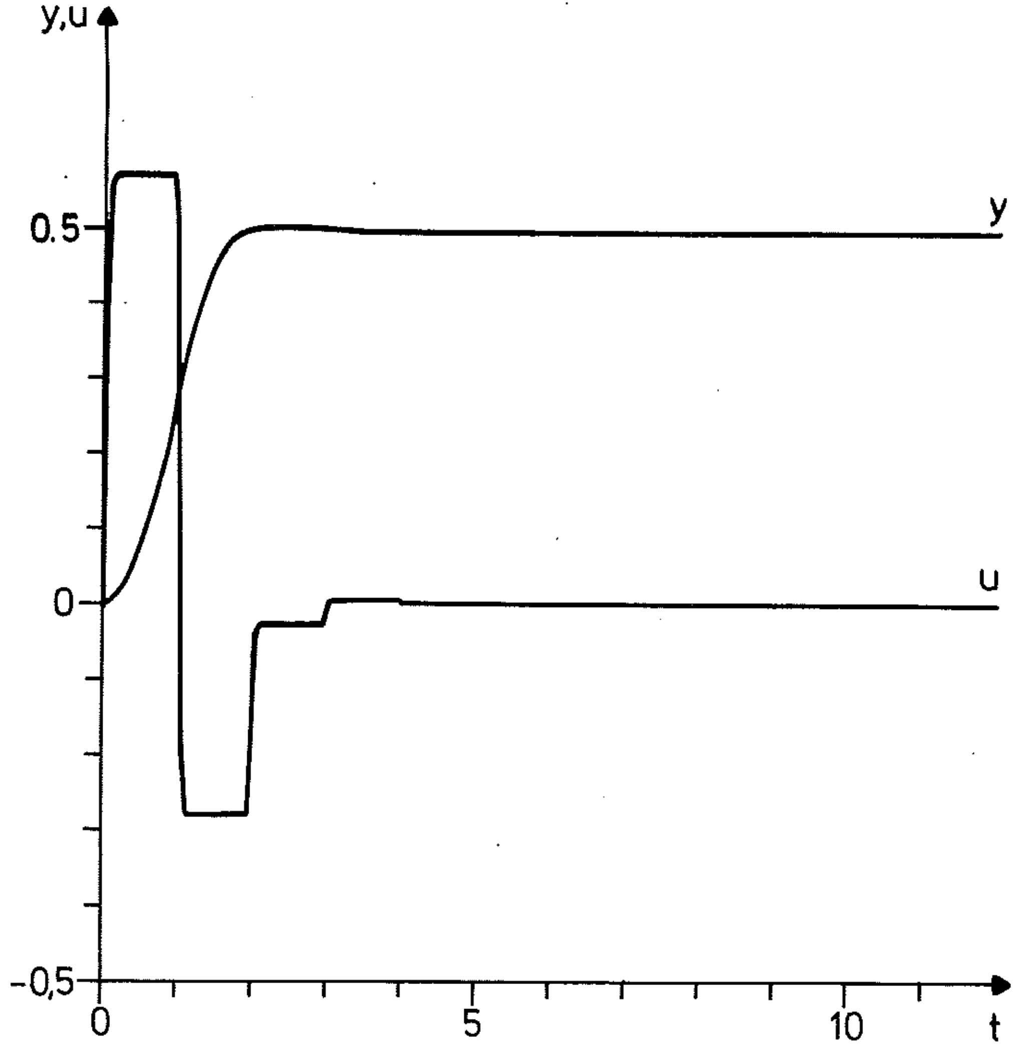
Poles in 0, -0.05 $\pm$ i0.15

Characteristic polynomial $q^3 + 0.1q^2 + 0.025q$

Regulator: $P^*(q^{-1}) = 1.286$

$R^*(q^{-1}) = 2.051 - 0.766q^{-1}$

$S^*(q^{-1}) = 1 + 0.574q^{-1}$

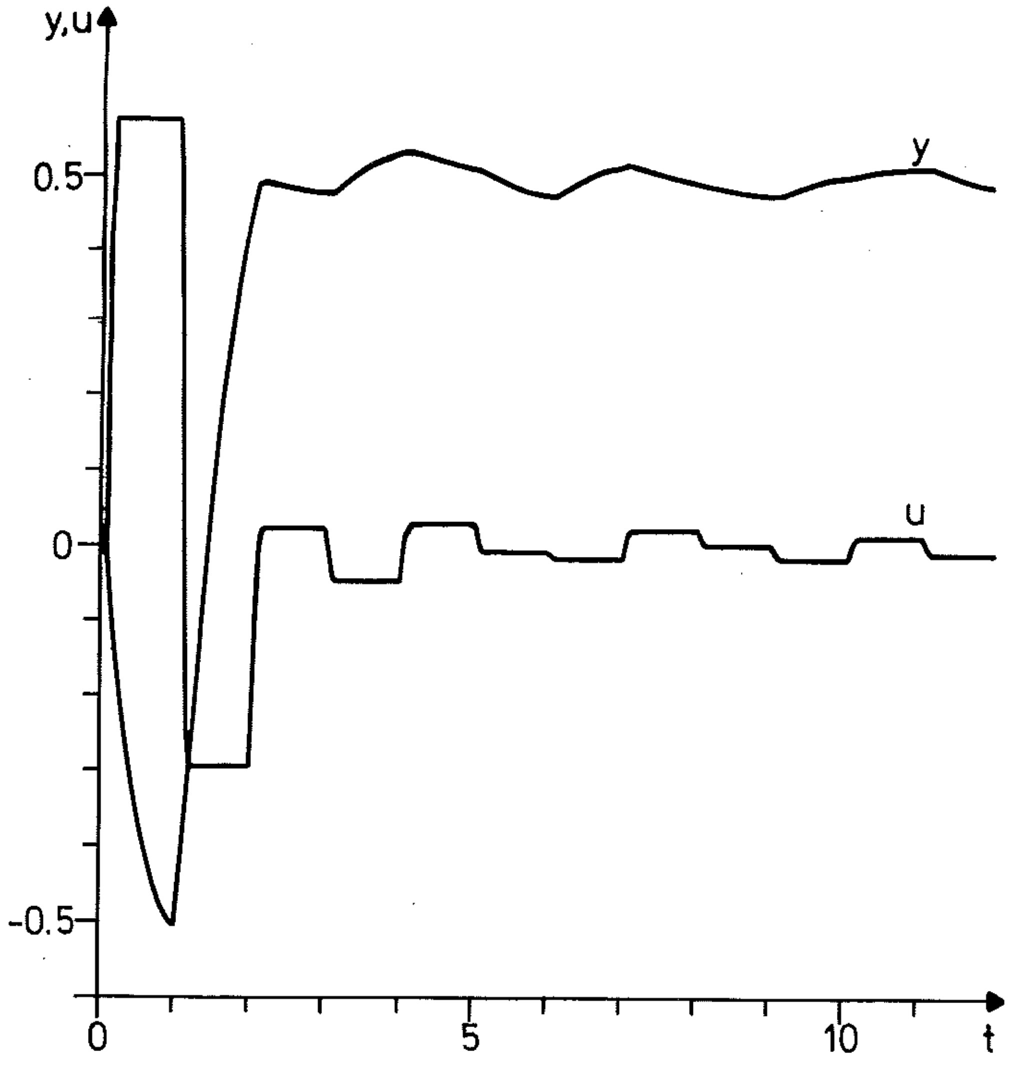


System 1. All poles in 0.

Regulator: $P^*(q^{-1}) = 1.144$

$R^*(q^{-1}) = 1.886 - 0.743q^{-1}$

$S^*(q^{-1}) = 1 + 0.557q^{-1}$

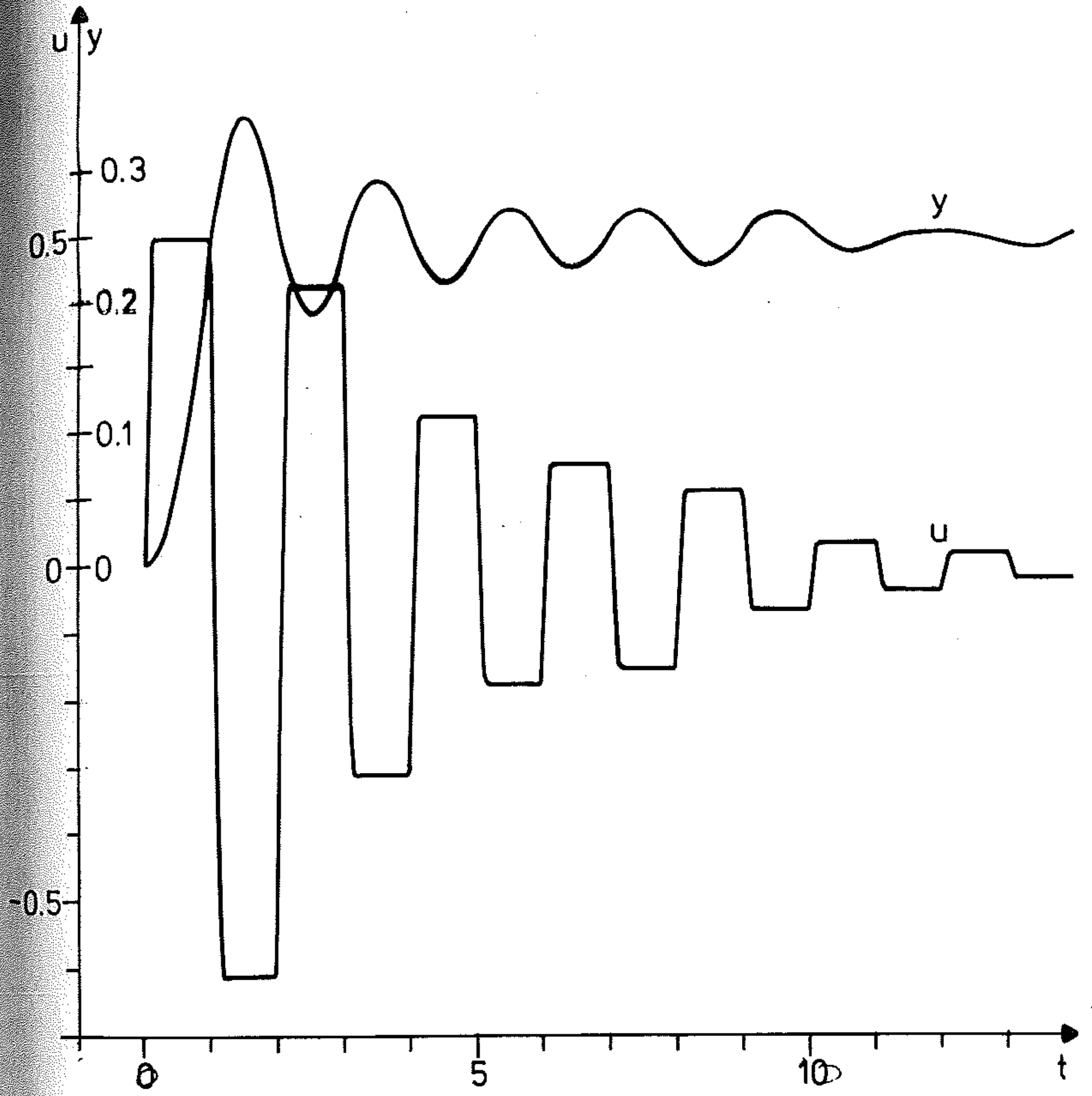


System 2. All poles in 0.

Regulator: $P^*(q^{-1}) = 1.144$

$R^*(q^{-1}) = 2.095 - 0.952q^{-1}$

$S^*(q^{-1}) = 1 + 3.333q^{-1}$



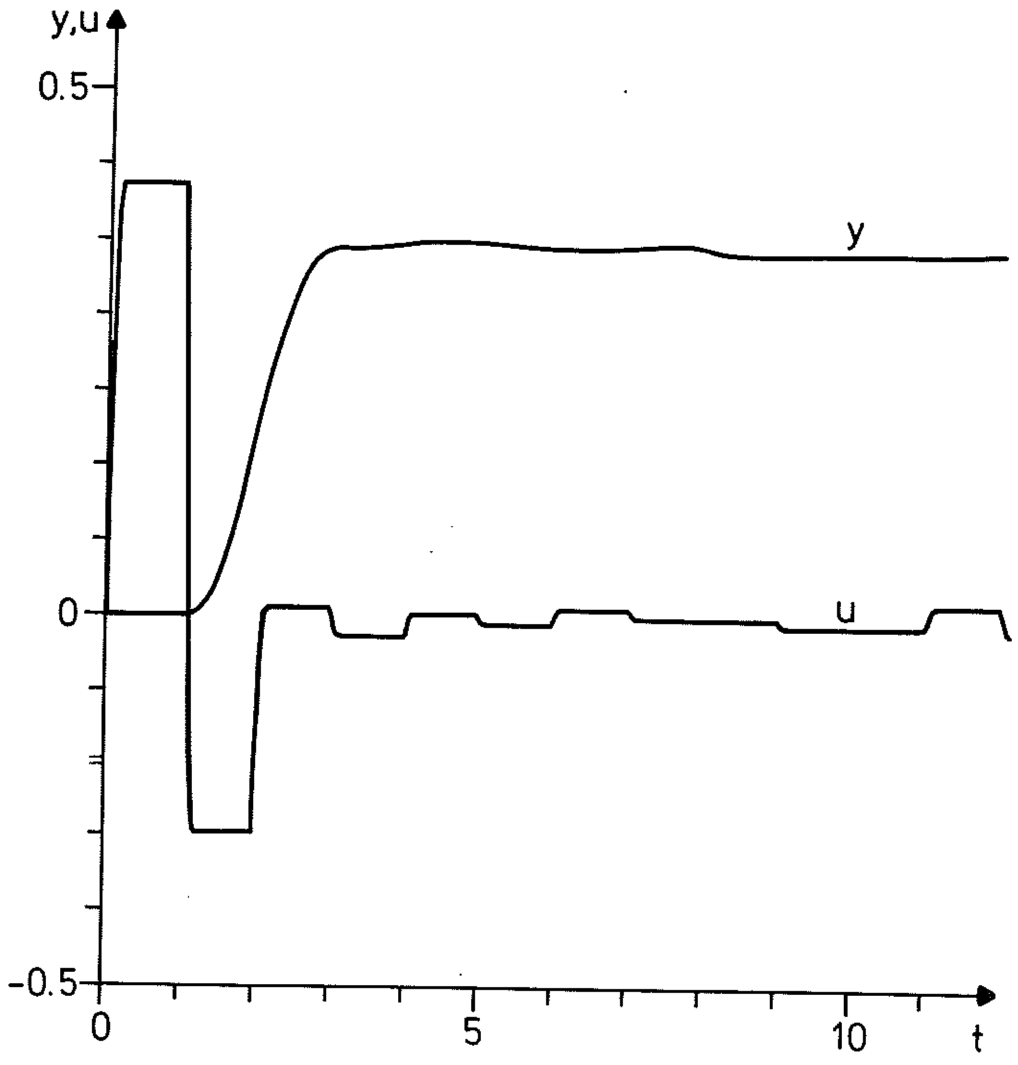
System 1. Dead beat, i.e. poles in -0.75, 0, 0.

Regulator: $P^*(q^{-1}) = 2.0$

$R^*(q^{-1}) = 3.0 - 1.0q^{-1}$

$S^*(q^{-1}) = 1.0 + 0.75q^{-1}$

Notice that in this case the step is only 0.25.



System 1 with one step time delay. All poles in 0.

Regulator: $P^{*}(q^{-1}) = 1.143$

$R^{*}(q^{-1}) = 2.06 - 0.943q^{-1}$

$S^{*}(q^{-1}) = 1 + 1.5q^{-1} + 0.707q^{-2}$

6. REFERENCES

[1] D Luenberger: An Introduction to Observers. IEEE Trans Automatic Control AC-16 (1971) No.6

[2] B van der Waerden: Algebra. Springer Verlag 1971

[3] K Jensen, N Wirth: PASCAL. User Manual and Report. Springer Verlag 1975

[4] Bishop, Parish, White: A Medium Level Programming Language for Microprocessors. Report RD/L/R 1882, Central Electricity Research Laboratories

[5] J E Aspenäs: Medium Level Programming Languages for Microprocessors. Report RE-177, May 1976, Dept of Automatic Control, Lund Institute of Technology

[6] L Andersson: Cross Assembly and Relocation of Programs for the Intel Microprocessors Using a PDP-15 as a Host Computer. Report 7602 Dept of Automatic Control, Lund Institute of Technology.

APPENDIX A

DISCO PROGRAM LISTS

Some of the programs are written in the programming language MLP. This language contains some of the normal high level statements like assignments, loops, if-statements etc, which need no explanation. However, the language also contains statements at a lower level, which may be more difficult to understand at a first glance. Some examples with explanations are given here.

REG B = ↑L-1 First decrement reg L, then use the contents of a L as an address and load B with the contents of this cell.

CALL SUB<...> In this case the statements within brackets are performed as a preparation before the subroutine call.

REG D↑ TEMP Register D is loaded with the address of the variable TEMP.

%= means "not equal to"

REG ↑C = D Use C as address and store the contents of D in this address.

```
;       PROGRAM DISCO
;       GENERAL SINGLE INPUT-SINGLE OUTPUT REGULATOR
;       AUTHOR LEIF ANDERSSON 1976-04-27
;
;       THE FOLLOWING COMMENTS DESCRIBE THE DATA BASE IN THE
;       NOTATION OF THE PROGRAMMING LANGUAGE PASCAL. THE TYPE "WORD" IS
;       CONSIDERED PREDECLARED. IT CONSISTS OF ONE EIGHT BIT WORD,
;       AND IT MAY REPRESENT DIFFERENT QUANTITIES AS DESCRIBED BELOW.
;       THE DOUBLE QUOTE, " , IS USED AS A COMMENT DELIMITER
;       WITHIN THE PASCAL NOTATION.
;
;
;TYPE   SINGLE=WORD
;         "THE TYPE SINGLE REPRESENTS A FIXED POINT,TWO'S COMPLEMENT
;          FRACTIONAL NUMBER WITH THE DECIMAL POINT IMMEDIATELY TO
;          THE RIGHT OF THE SIGN BIT"
;       TRIPLE=RECORD M1,M2,M3:WORD
;         "TRIPLE REPRESENTS A FIXED POINT, TWO'S COMPLEMENT REAL
;          NUMBER WITH THE DECIMAL POINT IMMEDIATELY TO THE RIGHT
;          OF THE MSB OF M2"
;       END;
;       FLOAT=  RECORD  EXP:INTEGER;
;                       MANT:TRIPLE
;       END;
;       VECTOR=RECORD   EXP:INTEGER
;                       V:ARRAY[0..7] OF SINGLE
;       END;
;
;VAR    YREF:SINGLE;      "REFERENCE VALUE. ADDRESS 0"
;       PNUM:INTEGER;     "LENGTH OF P. ADDRESS 1"
;       RNUM:INTEGER;     "LENGTH OF R. ADDRESS 2"
;       SNUM:INTEGER;     "LENGTH OF S. ADDRESS 3"
;       SWITCH:INTEGER;   "IF SWITCH=0 THEN RUN THE REGULATOR ELSE
;                          ZERO THE VECTORS YR,U,Y, THE TEMPORARIES
;                          TP,TR,TS AND THE ANALOG OUTPUT."
;       P:VECTOR;         "FEEDFORWARD POLYNOMIAL. ADDRESS 10"
;       R:VECTOR;         "FEEDBACK POLYNOMIAL. ADDRESS 20"
;       S:VECTOR;         "REGULATOR DENOMINATOR. ADDRESS 30"
;       YR:VECTOR;        "REFERENCE VALUES. ADDRESS 40"
;       Y:VECTOR;         "MEASURED VALUES. ADDRESS 50"
;       U:VECTOR;         "CONTROL VALUES. ADDRESS 60"
;       TP,TR,TS:FLOAT    "TEMPORARY STORAGE"
;
;END OF PASCAL NOTATION
;
;       SUBROUTINES REQUIRED:   SCAPR, FIX, VMOVE,FLOAT,ADSUB,
;                               SHIFT,LDST, MULT
;
;
;
;
;


        BANK 036
        WORDS YREF,PNUM,RNUM,SNUM,SWITCH
        WORDS P↑010,PV[7],R,RV[7],S,SV[7]
        WORDS YR,YRV[7],Y,YV[7],U,UV[7]
        WORDS TP[4],TR[4],TS[4]
        GLOBAL MULT,ADD3,SUBF,STORE4,RSUBF,FIX,VMOVE,SCAPR
        GLOBAL OPCOM,INIT
        SAME BANK
        BANK IS 036
;
```

```
      IF 0%=SWITCH THEN
        CALL INIT
      ELSE
;
;       COMPUTE THE LAST TERMS OF YR*P, SUBTRACT THE PREVIOUSLY
;       COMPUTED U*S.
;
        REG D=YREF
        YRV[0]=REG D
        CALL MULT<E=PV[0]>
        CALL ADD3<L↑TP[3]>
        REG B=↑L-1
        CALL SUBF<L↑TS>
        CALL STORE4<L↑TP>
;
;       READ PROCESS VALUE AND COMPUTE THE LAST TERM OF Y*R
;
        OUTPUT(22)=A;    START A/D CONVERSION
        REG D=RV[0]
        REG L↑YV[0]
        E=INPUT(2);      GET A/D VALUE
        REG ↑L=E
        CALL MULT
        CALL ADD3<L↑TR[3]>
        REG B=↑L-1
        CALL RSUBF<L↑TP>
        CALL FIX
        OUTPUT(23)=D;           WRITE ON D/A
        UV[0]=REG D
;
;       START COMPUTING FOR NEXT SAMPLE
;
        CALL VMOVE<E↑YRV[0],L↑PNUM>
        CALL VMOVE<E↑YV[0],L↑RNUM>
        CALL SCAPR<C↑SNUM,D↑S,E↑U>
        CALL STORE4<L↑TS>
        CALL SCAPR<C↑PNUM,D↑P,E↑YR>
        CALL STORE4<L↑TP>
        CALL SCAPR<C↑RNUM,D↑R,E↑Y>
        CALL STORE4<L↑TR>
        CALL VMOVE<E↑UV[0],L↑SNUM>
      END
      GOTO OPCOM
      FINISH
```

```
   PROC INIT
;  INITIALIZATION ROUTINE FOR DISCO
;
;  AUTHOR LEIF ANDERSSON 1976-04-26
;
;  THE ROUTINE WILL ZERO THE DATA BANK FROM YR AND UP (SEE
;  DISCO FOR A DESCRIPTION OF THE DATA BASE), THE EXPONENTS
;  OF THE TEMPORARIES TP, TR AND TS WILL THEN BE SET EQUAL TO
;  THE EXPONENTS OF P,R,S.
;
   WORDS SWITCH↑04,P↑010,R↑020,S↑030,YR↑040
   WORDS TP↑070,TR↑074,TS↑0100
   GLOBAL INIT
   SAME BANK
;
   REG L↑YR
   REG A=0
   WHILE A%=L DO
     REG ↑L=A
     REG L=L+1
   ENDWHILE
   OUTPUT(23)=0
   SWITCH=0
   TP=P
   TR=R
   TS=S
   ENDPROC
   FINISH
```

```
        PROC SCAPR
;       SCALAR PRODUCT OF TWO VECTORS IN THE SAME BANK
;       AUTHOR LEIF ANDERSSON 1975-11-20
;
;       ENTRY:  C↑ VECTOR LENGTH
;               D↑ FIRST VECTOR
;               E↑ SECOND VECTOR
;       EXIT:   B=EXPONENT OF RESULT
;               CDE=MANTISSA OF RESULT
;       REGISTERS AFFECTED
;       HIGHEST FREE CELL: 367
;       SUBROUTINES REQUIRDED
;               MULT
;               ADD3
;               LOAD4
;               STORE3
;               STORE4
;
        WORDS COUNT[4]↑0370,TEMP[4]
        GLOBAL LOAD4,STORE3,STORE4,ADD3,MULT
        GLOBAL SCAPR
        SAME BANK
;
;       ZERO THE TEMPORARY STORAGE
;
        REG B=0
        REG L↑TEMP
        PUSH ↑L+1=B,B,B,B
;
;       TEST FOR ZERO LENGTH
;
        REG C=↑C
        IF 0%=C THEN
;
;         COMPUTE AND STORE EXPONENT
;
          REG A=↑D
          REG L=E
          TEMP=A+M
;
          REPEAT
;
;           INCREMENT AND SAVE COUNT AND POINTERS
;
            REG B=B+1
            REG D=D+1
            REG E=E+1
            CALL STORE4<L↑COUNT>
;
            CALL MULT<D=↑D,E=↑E>
            CALL ADD3<L↑TEMP[3]>
            CALL STORE3
            CALL LOAD4<L↑COUNT>
          UNTIL B=C
        END
        CALL LOAD4<L↑TEMP>
        ENDPROC
        FINISH
        .TITLE    FIX 002
/       SUBROUTINE FIX
/       CONVERTS A FLOATING POINT NUMBER TO A FIXED POINT
/       FRACTIONAL ONE WORD NUMBER. IF THE PROCESS GIVES
/       OVERFLOW THE RESULT IS SET PLUS OR MINUS FULL SCALE.
/       AUTHOR LEIF ANDERSSON 1974-11-20
/       REVISED LEIF ANDERSSON 1975-01-02
/
/       ENTRY:  B = EXPONENT
/               CDE = MANTISSA
/       EXIT:   A = SIGN OF NUMBER (0 OR -1)
/               D = RESULT
/               L = SIGN OF NUMBER
/       REGISTERS AFFECTED: A,B,C,D,E,L
/       HIGHEST FREE CELL: 377
/       SUBROUTINES REQUIRED
/               SHIFT
/
        .EJECT
        .GLOBL FIX
        .GLOBL RSHIFT,LSHIFT
/
/SET L=0 OR -1 DEPENDING ON THE SIGN OF THE MANTISSA
/
FIX     LAC
        RAL
        SBA
        LLA
/
/CHECK IF RIGHT OR LEFT SHIFT
/
        XRA
        CPB
        JTZ ROUND
        JFS RLOOP
/
/LEFT SHIFT. CHECK FOR OVERFLOW IN EACH STEP.
/IF C <>L OVERFLOW HAS OCCURED.
/
        LAC
LLOOP   CPL     /A=C ON RETURN FROM LSHIFT
        JFZ OFLO
        CAL LSHIFT
        JFZ LLOOP       /THE FLAGS ARE SET BY DCB IN LSHIFT
/
/RIGHT SHIFT
/
RLOOP   CFZ RSHIFT
        JFZ RLOOP       /THE FLAGS ARE SET BY INB IN RSHIFT
/
/ROUND OFF C AND D
/
ROUND   LAE
        RAL
        LAB     /B CONTAINS 0
        ACD
        LDA
        LAB
        ACC
        LCA
/
```

```
/CHECK FOR OVERFLOW. IF BOTH C=0 AND D=0 THE RESULT IS
/ZERO REGARDLESS OF THE VALUE OF L
/
        LAC
        ORD
        RTZ
/
/IF C <> L OVERFLOW HAS OCCURED
/
        LAC
        CPL
        JFZ OFLO
/
/CHECK THE SIGN BIT OF D
/
        LAD
        XRL
        RFS
/
/OVERFLOW
/
OFLO    LAI 177
        SUL
        LDA
        RET
        .END
```

```
    PROC VMOVE
;   MOVES A VECTOR ONE STEP UPWARDS IN MEMORY. THE TOP ELEMENT
;   IS LOST AND THE BOTTOM CELL IS ZEROED.
;
;   AUTHOR LEIF ANDERSSON 1975-12-05
;
;   ENTRY:   E↑ BOTTOM ELEMENT
;            L↑ VECTOR LENGTH
;   EXIT:    L↑ BOTTTOM ELEMENT
;   REGISTERS AFFECTED: A,E,L
;   SUBROUTINES REQUIRED
;            NONE
;
    GLOBAL VMOVE
;
    REG A=M
    RETURN IF A=0
    REG L=E
    REG E=A
    L=A+L
    REG L=L-1
    LOOP
      REG E=E-1
    EXIT IF ZERO TRUE
      REG A=↑L-1
      REG ↑L+1=A
      REG L=L-1
    ENDLOOP
    REG M=0
    ENDPROC
    FINISH
```

```
;       SUBROUTINE ADDF
;       SUBROUTINE SUBF
;       SUBROUTINE RSUBF
;       FLOATING POINT ADD,SUBTRACT AND REVERSE SUBTRACT
;       AUTHOR LEIF ANDERSSON 1975-11-27
;
;       ENTRY:  BCDE=FIRST TERM
;               L↑ SECOND TERM
;       EXIT:   BCDE=RESULT
;       REGISTERS AFFECTED: A,B,C,D,E,L
;       HIGHEST FREE CELL: 373
;       SUBROUTINES REQUIRED
;               LOAD4
;               STORE4
;               RSHIFT
;               ADD3
;               SUB3
;               RSUB3
;
        WORDS TEMP[4]↑0374
        SAME BANK
        GLOBAL LOAD4,STORE4,RSHIFT,ADD3,SUB3,RSUB3
        GLOBAL ADDF,SUBF,RSUBF
;
        PROC ADDF
        CALL ASSET
        CALL ADD3
        ENDPROC
        PROC SUBF
        CALL ASSET
        CALL RSUB3
        ENDPROC
        PROC RSUBF
        CALL ASSET
        CALL SUB3
        ENDPROC
;
        PROC ASSET
;       LOCAL PROCEDURE TO PERFORM THE NECESSARY SHIFTS.
;       ASSET WILL PUT THE SHIFTED FIRST TERM IN TEMP AND LEAVE
;       THE SHIFTED SECOND TERM IN THE REGISTERS
;
        WHILE B<M DO
          CALL RSHIFT
        ENDWHILE
        REG A=L
        CALL STORE4<L↑TEMP>
        CALL LOAD4<L=A>
        REG L↑TEMP
        WHILE B%=M DO
          CALL RSHIFT
        ENDWHILE
        REG L↑TEMP[3]
        ENDPROC
        FINISH
```

```
/       SUBROUTINES ADD3, SUB3
/       ADDS OR SUBTRACTS THREE-WORD NUMBERS
/       AUTHOR LEIF ANDERSSON 1974-02-06
/
/       ENTRY:  FIRST TERM IN C,D,E
/               ADDRESS OF SECOND TERM IN H,L
/       EXIT:   RESULT IN C,D,E
/               ADDRESS OF LAST WORD OF SECOND TERM IN H,L
/       REGISTERS AFFECTED: A,C,D,E,L
/       SUBROUTINES REQUIRED
/               NONE
/
        .GLOBL ADD3,SUB3,RSUB3
ADD3    LAE
        ADM
        LEA
        DCL
        LAD
        ACM
        LDA
        DCL
        LAC
        ACM
        LCA
        RET
SUB3    LAE
        SUM
        LEA
        DCL
        LAD
        SBM
        LDA
        DCL
        LAC
        SBM
        LCA
        RET
RSUB3   LAM
        SUE
        LEA
        DCL
        LAM
        SBD
        LDA
        DCL
        LAM
        SBC
        LCA
        RET
        .END
```

```
/       SUBROUTINES RSHIFT, LSHIFT
/       SHIFTS A THREE-WORD MANTISSA ONE STEP RIGHT OR LEFT AND
/       CHANGES THE EXPONENT ACCORDINGLY.
/       AUTHOR LEIF ANDERSSON 1974-03-04
/
/       ENTRY:  EXPONENT IN B,
/               MANTISSA IN C,D,E.
/       EXIT:   EXPONENT IN B,
/               MANTISSA IN C,D,E
/       REGISTERS AFFECTED: A,B,C,D,E
/       SUBROUTINES REQUIRED
/               NONE
/
        .GLOBL RSHIFT,LSHIFT
RSHIFT  LAC
        RAL     /SIGN BIT INTO CARRY
        LAC
        RAR
        LCA
        LAD     /SHIFT D
        RAR
        LDA
        LAE     /SHIFT E
        RAR
        LEA
        INB     /INCREMENT EXPONENT
        RET
LSHIFT  XRA     /ZERO CARRY
        LAE     /SHIFT E
        RAL
        LEA
        LAD     /SHIFT D
        RAL
        LDA
        LAC     /SHIFT C
        RAL
        LCA
        DCB     /DECREMENT EXPONENT
        RET
        .END
```

```
/         SUBROUTINES LOAD4, LOAD3, STORE4, STORE3
/         LOADS OR STORES B,C,D,E OR C,D,E
/         AUTHOR LEIF ANDERSSON 1974-02-06
/
/         ENTRY:  ADDRESS OF FIRST WORD IN H,L
/         SUBROUTINES REQUIRED
/                 NONE
/
          .GLOBL LOAD4,LOAD3,STORE4,STORE3
LOAD4     LBM
          INL
LOAD3     LCM
          INL
          LDM
          INL
          LEM
          RET
STORE4    LMB
          INL
STORE3    LMC
          INL
          LMD
          INL
          LME
          RET
          .END
```

```
        .TITLE    MULT 001
/       SUBROUTINE MULT
/       MULTIPLIES EIGHT-BIT, TWO'S COMPLEMENT FRACTIONAL NUMBERS
/       AUTHOR LEIF ANDERSSON 1974-03-19
/
/       ENTRY:  NUMBERS IN D AND E
/       EXIT:   RESULT IN D AND E. C IS 0 IF RESULT IS
/               POSITIVE AND -1 IF RESULT IS NEGATIVE.
/       REGISTERS AFFECTED: A,B,C,D
/       SUBROUTINES REQUIRED
/               NONE
/
        .EJECT
        .GLOBL MULT
/SETUP PHASE
/CHECK IF RESULT IS ZERO. IF SO SET RESULT AND RETURN
MULT    XRA
        CPD
        JTZ ZERO        /IF D .EQ. 0 GO TO ZERO
        CPE
        JFZ NOZER       IF E .NE. 0 GO TO NOZER
ZERO    LCA     /C=0
        LDA     /D=0
        LEA     /E=0
        RET
/COMPUTE SIGN OF RESULT. STORE IN B.
NOZER   LBA
        LCA
        LAD
        XRE
        RAL
        LAB
        SBB
        LBA
/REPLACE NUMBERS WITH MAGNITUDES.
        LAC     /A=0
        SUD     /A=-D
        JTS DOK /IF D >=0 GO TO DOK
        LDA     /D=-D
DOK     LAC
        SUE
        JTS STEPS
        LEA
/SHIFT AND ADD USING SUBROUTINE STEP
STEPS   XRA     /A=0
        CAL STEP
        CAL STEP
        CAL STEP
        CAL STEP
        CAL STEP
        CAL STEP
        CAL STEP
        CAL STEP
/MAGNITUDE OF RESULT IS IN A AND D. MOVE TO D AND E
        LED
        LDA
/CHANGE SIGN IF NECESSARY. RETURN
        LCB
        XRA
        LBA
        CPC
        RTZ
        SUE
        LEA
        LAB
        SBD
        LDA
        RET      /RETURN
/
/SUBROUTINE STEP
/
/SHIFT A AND D ONE STEP RIGHT
STEP    RAR
        LCA
        LAD
        RAR
        LDA
/IF CARRY =1 THEN ADD, ELSE RETURN
        LAC
        RFC
        ADE
        RET
        .END
```

APPENDIX B

OPCOM Program Lists

```
/          ROUTINE OPCOM
/
/          DRIVE ROUTINE FOR THE OPERATORS CONSOL
/          USED WITH THE MICRO COMPUTER INTEL 8008.
/
/          AUTHOR  HILDING ELMQVIST  1973-10-16
/
/          SUBROUTINE REQUIRED
/                    OB
/                    INTDB
/                    FRACDB
/                    BO
/                    INTBD
/                    FRACBD
/
        .GLOBL   OPCOM
        .GLOBL   OB,BO,INTBD,INTDB,FRACDB,FRACBD
/
RHADR=117
RLADR=115
RSW=113
RDAT1=111
RDAT2=107
/
WDIS1=163
WDIS2=161
/
SAVE=200
RAMBNK=37
/
/
OPCOM   LHI      RAMBNK
        RSW
        RRC                / CARRY = IN-BUTTON
        JTC      DISP
/
/
/==============================================================================
/
/
/          INPUT SECTION
        RDAT1
        LLI      SAVE
        LMA                / SAVE SIGN
        NDI      017       / MASK OUT 1:ST DIGIT
        LBA                / B = 1:ST DIGIT
        RDAT2
        LDA
        NDI      360       / MASK OUT 2:ND DIGIT
        RRC
        RRC
        RRC
        RRC
        LCA                / C = 2:ND DIGIT
        LAD
        NDI      017       / MASK OUT 3:RD DIGIT
        LDA                / D = 3:RD DIGIT
/
        RSW
        RRC
        RRC                / CARRY = OCT - DEC SWITCH
        JFC     OCT1
/
/         DECIMAL
        RRC                     / CARRY = INT - FRAC SWITCH
        JFC      INT1
        JMP     FRAC1
/
OCT1    CAL OB
        JMP     SET
INT1    CAL INTDB
        JMP     SET
FRAC1   CAL FRACDB
/
/
SET     LEA                     / E = BINARY VALUE
        LHI     RAMBNK
        LLI     SAVE
        XRA                     / A = 0
        ADM                     / GET SIGN
        JFS     ADR
        XRA
        SUE                     / CHANGE SIGN
        LEA
/
ADR     RHADR                   / GET ADDRESS
        LHA
        RLC                     / ERROR ?
        JFC     OPCOM
        RLADR
        LLA
        LME                     / ASSIGN
        LHI     RAMBNK          / RESET HIGH ADDRESS
        JMP     OPCOM
/
/
/=====================================================================
/
/
/         DISPLAY SECTION
DISP    RHADR                   / GET ADDRESSS
        LHA
        RLC
        JFC     ERROR
        RLADR
        LLA
/
        LEI     000
        RSW
        RRC
        RRC                     / CARRY = OCT - DEC SWITCH
        JFC     OCT2
/
/         DECIMAL
        RRC                     / CARRY = INT - FRAC SWITCH
        JFC      INT2
        JMP     FRAC2
/
/
OCT2    RRC                     / CARRY = INT - FRAC SWITCH
        JFC     01
        ORE                     / LIGHT .
```

```
01      RRC             / CARRY = SIGNED - UNSIGNED SWITCH
        LAM
        CTC     SIGN
        LLE             / SAVE E IN L
        CAL     B0
        LAL
        JMP     PACK
/
/
INT2    CAL     SIGN
        CAL     INTBD
        LAE
        JMP     PACK
/
/
FRAC2   LAM             / TEST IF BINARY VALUE = 200 (-1.000)
        CPI     200
        JFZ     F1
        LBI     160     / LIGHT -1.
        LCI     000
        JMP     DISPL
/
F1      CAL     SIGN
        LLE
        CAL     FRACBD
        LHI     RAMBNK  / RESET HIGH ADDRESS
        LAL
        ORI     020     / LIGHT .
        JMP     PACK
/
/
PACK    ORB
        LBA             / B = F B
        LAC
        RLC
        RLC
        RLC
        RLC
        ORD
        LCA             / C = C D
        JMP     DISPL
/
/
ERROR   LBI     017
        LCI     377
/
/
DISPL   LAB
        WDIS1
        LAC
        WDIS2
        JMP     OPCOM
/
/
/------------------------------------------------------------------------
/
/
SIGN    XRA             / GET BINARY VALUE
        ADM
        JTS     NEG
/
```

```
/           POSITIVE
        LAE
        ORI     300     / LIGHT +
        LEA
        LAM
        RET
/
/           NEGATIVE
NEG     LAE
        ORI     100     / LIGHT -
        LEA
        XRA             / A=0
        SUM             / CHANGE SIGN
        RET
/
        .END
```

```
/            SUBROUTINE BO
/
/            CONVERTS BINARY VALUE TO OCTAL DIGITS.
/
/            AUTHOR  HILDING ELMQVIST  1973-10-16
/
/            A      - BINARY VALUE (INPUT)
/            B      - RETURNED 1:ST DIGIT
/            C      - RETURNED 2:ND DIGIT
/            D      - RETURNED 3:RD DIGIT
/            E      - INTERNAL USE
/
/
          .GLOBL  BO
BO        LEA             / E = A
          NDI     300     / GET 1:ST DIGIT
          RLC             / SHIFT
          RLC
          LBA             / B = 1:ST DIGIT
/
          LAE
          NDI     070     / GET 2:ND DIGIT
          RRC             / SHIFT
          RRC
          RRC
          LCA             / C = 2:ND DIGIT
/
          LAE
          NDI     007     / GET 3:RD DIGIT
          LDA             / D = 3:RD DIGIT
          RET
/
          .END
```

```
/           SUBROUTINE OB
/
/           CONVERTS OCTAL DIGITS TO BINARY VALUE
/
/           AUTHOR  HILDING ELMQVIST  1973-10-16
/
/           A       - RETURNED BINARY VALUE
/           B       - 1:ST DIGIT (INPUT)
/           C       - 2:ND DIGIT (INPUT)
/           D       - 3:RD DIGIT (INPUT)
/           E       - INTERNAL USE
/
        .GLOBL OB
/
OB      LAB                 / A = 1:ST DIGIT
        RRC
        RRC
        NDI     300         / A7,A6 = 1:ST DIGIT
        LEA
        LAC                 / A = 2:ND DIGIT
        RLC
        RLC
        RLC
        NDI     070         / A5,A4,A3 = 2:ND DIGIT
        ORE
        LEA
        LAD                 / A = 3:RD DIGIT
        NDI     007         / A2,A1,A0 = 3:RD DIGIT
        ORE
        RET
/
        .END
```

```
/       SUBROUTINE FRACBD
/
/         CONVERTS BINARY VALUE CONSIDERED AS FRACTIONAL NUMBER
/         TO DECIMAL DIGITS
/
/         AUTHOR LENNART NILSSON 06.05.73
/         REVISED HILDING ELMQVIST 1973-10-17
/
/       ENTER WITH X IN REG A
/
/       EXIT WITH DECIMAL DIGITS IN REG AS FOLLOWS
/       X = .BCD
/
/       REGISTERS AFFECTED A,B,C,D,E,H
/
        .GLOBL  FRACBD
FRACBD  LBI     0
        LCB
        LDB
        LEB
        RLC
        RLC             / CHECK SECOND BIN. DIGIT
        JFC     NUL2
        LBI     005     / IF.EQ.1 B=B+5
NUL2    RLC             / CHECK DIGIT 3
        JFC     NUL3
        LCI     005     / IF.EQ.1 B=B+2, C=C+5
        INB
        INB
NUL3    RLC             / CHECK DIGIT 4
        JFC     NUL4
        LDI     005     / IF.EQ1 B=B+1, C=C+2, D=D+5
        INB
        INC
        INC
NUL4    RLC             / CHECK DIGIT 5
        JFC     NUL5
        LEI     005     / IF.EQ.1 C=C+6, D=D+2, E=E+5
        LHA
        LAI     006
        ADC
        LCA
        IND
        IND
        LAH
NUL5    RLC             / CHECK DIGIT 6
        JFC     NUL6
        INC             / IF.EQ.1 C=C+3, D=D+1, E=E+3
        INC
        INC
        IND
        INE
        INE
        INE
NUL6    RLC             / CHECK DIGIT 7
        JFC     NUL7
        INC             / IF.EQ.1 C=C+1, D=D+5, E=E+6
        LHA
        LAI     005
        ADD
        LDA
```

```
          LAI     006
          ADE
          LEA
          LAH
NUL7      RLC             / CHECK DIGIT 8
          JFC     NUL8
          LHA             / IF,EQ,1 D=D+7, E=E+8
          LAI     007
          ADD
          LDA
          LAI     010
          ADE
          LEA
NUL8      LAE             / REG E=E+5 FOR CORRECT LAST
          ADI     005     / DECIMAL DIGIT
REGE      CPI     012     / CHECK IF E,GT,10
          JTS     REGD
          SUI     012     / IF,GT,10 E=E-10, D=D+1
          IND
          JMP     REGE    / DO IT AGAIN
REGD      LEA
          LAD
LOOPD     CPI     012     / CHECK IF D,GT,10
          JTS     REGC
          SUI     012     / IF,GT,10 D=D-10, C=C+1
          INC
          JMP     LOOPD   / DO IT AGAIN
REGC      LDA
          LAC
          CPI     012     / CHECK IF C,GT,10
          JTS     READY
          SUI     012     / IF,GT,10 C=C-10, B=B+1
          INB
          LCA
READY     XRA             / REG A=0
RETURN    RET
          .END
```

```
/        SUBROUTINE FRACDB
/
/          CONVERTS DECIMAL DIGITS TREATED AS FRACTIONAL NUMBER
/          TO BINARY VALUE.
/
/          AUTHOR LENNART NILSSON 08.05.73
/          REVISED HILDING ELMQVIST 1973-10-17
/
/          ENTER WITH DECIMAL DIGITS IN REGISTER
/          AS FOLLOWS X=.BCD
/
/          EXIT WITH BINARY CODED X IN REG A
/
/          REGISTERS AFFECTED A,B,C,D,E,H
/
         .GLOBL  FRACDB
FRACDB   XRA                 / PUT ZERO IN
         LHA                 / ADD-REG E AND H
         LEA
DIG1     DCB                 / FIRST DEC. DIGIT
         JTS     DIG2        / TEST IF DIGIT WAS ZERO
         LAI     314         / ADD DEC. 0.1 IN DOUBLE PREC TO E,H
         ADH
         LHA
         LAI     014
         ACE
         LEA
         JMP     DIG1        / DO IT AGAIN
DIG2     DCC                 / SECOND DEC. DIGIT
         JTS     DIG3        / TEST IF DIGIT WAS ZERO
         LAI     107         / ADD DEC. 0.01 TO E AND H
         ADH
         LHA
         LAI     001
         ACE
         LEA
         JMP     DIG2        / DO IT AGAIN
DIG3     DCD                 / THIRD DIGIT
         JTS     READY       / TEST IF DIGIT WAS ZERO
         LAI     040         / ADD DEC 0.001 TO E AND H
         ADH
         LHA
         LAI     000
         ACE
         LEA
         JMP     DIG3
READY    LAH                 /  GET MOST SIGNIFICANT DIGIT FROM REG. H
         RLC
         LAI     000
         ACE                 / ADD WITH CARRY
         RET
         .END
```

```
/        SUBROUTINE INTBD
/
/          CONVERTS BINARY VALUE TREATED AS INTEGER TO DECIMAL DIGITS
/
/          AUTHOR   LENNART NILSSON 08.05.73
/          REVISED HILDING ELMQVIST 1973-10-17
/
/        ENTER WITH X IN REG A
/
/        EXIT WITH DECIMAL DIGITS IN REG
/        AS FOLLOWS: X = BCD
/
/        REGISTERS AFFECTED A,B,C,D
/
         .GLOBL   INTBD
INTBD    LBI      000       / B = 0
         LCB                / C = 0
REGD     CPI      012
         JTS      REGC
         SUI      012
         INC
         JMP      REGD
/
/          C = X/012  ;  A = X-C*012
REGC     LDA
         LAC
CLOOP    CPI      012
         JTS      READY
         SUI      012
         INB
         JMP      CLOOP
/
/          B = C/012  ;  A = C-C*012
READY    LCA
         RET
         .END
```

```
/         SUBROUTINE INTDB
/
/           CONVERTS DECIMAL DIGITS TREATED AS INTEGER TO BINARY VALUE
/
/           AUTHOR LENNART NILSSON 08.05.73
/           REVISED HILDING ELMQVIST  1973-10-17
/
/         ENTER WITH DECIMAL DIGITS IN REGISTER AS
/         FOLLOWS X = BCD
/
/         EXIT WITH BINARY CODE IN REG A
/
/         REGISTERS AFFECTED A,B,C,D
/
          .GLOBL  INTDB
INTDB     XRA
DIG1      DCB
          JTS     DIG2
          ADI     144
          JMP     DIG1
DIG2      DCC
          JTS     DIG3
          ADI     012
          JMP     DIG2
DIG3      DCD
          JTS     READY
          ADI     001
          JMP     DIG3
READY     RET
          .END
```

APPENDIX C

A SUMMARY OF THE THEORY FOR REDUCED ORDER STATE OBSERVERS

Consider a completely observable system

$$x(t+1) = \phi\, x(t) + \Gamma\, u(t)$$

$$y(t) = \Theta\, x(t)$$

where ϕ is n×n, Γ is n×1 and Θ is 1×n. Assume that Θ is of the form $[\,1 \quad 0 \quad 0 \ldots 0\,]$. If this is not the case originally, a simple transformation will bring the system to the desired form. Partition the state vector into x_1 of length 1 and x_2 of length n-1:

$$\begin{pmatrix} x_1(t+1) \\ x_2(t+1) \end{pmatrix} = \begin{pmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{pmatrix} \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \end{pmatrix} u(t)$$

$$y(t) = \begin{pmatrix} 1 & 0 & \ldots & 0 \end{pmatrix} x$$

Since $x_1 = y$, a full order observer for x_2 will also constitute a reduced order observer for x. The following lemma will be needed:

Lemma. If the pair $[\phi,\Theta]$ is completely observable, then so is the pair $[\phi_{22},\phi_{12}]$.

Proof. Let $x_1(0) = 0$ and $u(t) = 0$. If $[\phi_{22},\phi_{12}]$ is not completely observable, there exists an initial value $x_2(0) = x_{20}$ such that $\phi_{12}\, x_2(t) = 0$ for all t. But then $x_1(t) = y(t) = 0$ for all t, which implies that $[\phi,\Theta]$ has an unobservable state which is a contradiction.

A direct approach for the observer gives:

$$\hat{x}_2(t+1) = \phi_{21}\, y(t) + \phi_{22}\, \hat{x}_2(t) + \Gamma_2\, u(t) +$$
$$+ K[y(t+1) - \phi_{11}\, y(t) - \phi_{12}\, \hat{x}_2(t) - \Gamma_2\, u(t)]$$

Introduce

$$z(t) = x_2(t) - K\,y(t)$$

$$\hat{z}(t) = \hat{x}_2(t) - K\,y(t)$$

$$\tilde{x}_2(t) = x_2(t) - \hat{x}_2(t) = z(t) - \hat{z}(t)$$

$$\hat{z}(t+1) = [\phi_{22}-K\phi_{12}]\,\hat{x}_2(t) + [\phi_{21}-K\phi_{11}]\,y(t) - [\Gamma_2-K\Gamma_1]\,u(t) =$$

$$= [\phi_{22}-K\phi_{12}]\,\hat{z}(t) + [\phi_{21}-K\phi_{11} + \phi_{22}K-K\phi_{12}K]\,y(t) +$$

$$+ [\Gamma_2-K\Gamma_1]\,u(t)$$

$$\hat{x}_2(t) = \hat{z}(t) + K\,y(t)$$

The reconstruction error is given by

$$\tilde{x}(t) = z(t) - \hat{z}(t)$$

$$\tilde{x}(t+1) = [\phi_{22} - K\phi_{12}]\,\tilde{x}(t)$$

Since the pair $[\phi_{22},\phi_{12}]$ is completely observable, the eigenvalues of $[\phi_{22}-K\phi_{12}]$ may be arbitrarily chosen.

The reduced order observer is thus given by the dynamical system

$$\hat{z}(t+1) = \phi_r\,z(t) + \Gamma_{ry}\,y(t) + \Gamma_{ru}\,u(t)$$

$$\hat{x}_2(t) = \hat{z}(t) + K\,y(t)$$

with

$$\phi_r = \phi_{22} - K\phi_{12}$$

$$\Gamma_{ry} = \phi_{21} - K\phi_{11} + \phi_{22}K - K\phi_{12}K$$

$$\Gamma_{ru} = [\Gamma_2 - K\Gamma_1]$$