Engineering Research Institute

UNIVERSITY OF MASSACHUSETTS

AMHERST, MASSACHUSETTS

Project NGR-22-010-018

Report No. 4

A DIGITAL COMPUTER PROGRAM FOR CONTROL SYSTEM ANALYSIS

B. W. Lovell

August, 1967

PROGRAM LOVELL

A DIGITAL COMPUTER PROGRAM FOR CONTROL SYSTEM ANALYSIS

By

B. W. Lovell

August 17, 1967

TABLE OF CONTENTS

ABSTRACT

Program LOVELL is a digital computer program for the analysis of control systems. The program will handle PFM units, PWM units, zero order sample and hold units, and delay functions, as well as simple continuous systems.

The user of program LOVELL must supply the system equations to the program through a subroutine using state variable techniques. All of the differential equations are solved by a fourth order Runge-Kutta method, with the basic mesh size supplied by the user.

Program LOVELL has the capability of restarting a solution, or starting a solution, at a time other than zero. Also, through a subroutine, problems may be stacked. This allows more than one problem to be solved with only one computation, thus saving computer time.

This paper includes numerous sample problems. The data and necessary subroutines are shown for each problem.

The program is written in CDC 3600 Fortran.

INTRODUCTION

Program LOVELL uses a state variable approach for analyzing control systems of great diversity. While designed primarily for pulse frequency modulator (PFM) systems with a delay in the feedback loop, it will handle any of the following:

- Simple continuous systems
- Continuous systems with delayed functions (transport lags)
- Sampler-clamper systems
- Sampler-clamper systems with delayed functions
- Simple PFM systems
- PFM systems with sampler-clampers and/or delayed functions and/or a delayed pulse from one PFM
- Simple pulse width modulator (PWM) systems
- PWM systems with sampler-clampers and/or delayed functions and/or PFMs.

The PFMs may exhibit input saturation if desired, i.e., there may be a minimum pulse spacing. They may be discrete pulse frequency modulators (DPFM) if desired. The PFMs normally have impulses as outputs, but one PFM may be modified to have finite width output pulses. One PFM may directly feed another.

The PFMs may be coupled; the general equation describing the integrator of a PFM is

$\dot{p}$ = f(state variables of continuous system, integrators of PFMs, clamped quantities, output pulses of PFMs, delayed functions, time).

The IPFM, NPFM, and ΣPFM are all special cases.

It is possible to modify the characteristics of a PFM to produce a sign function generator (ideal contactor without deadband).

The program will handle up to ten PFM units, ten sampler-clampers, five delayed functions, and one PWM or one delayed pulse. The number of state variables of the continuous system plus the number of PFMs must be no greater than forty.*

----

*It is a trivial matter to increase the number of anything except PWMs or pulse delays.

The equations may be linear or nonlinear, time dependent or time invariant. All differential equations are solved by a fourth order Runge-Kutta method. The basic mesh size or calculation interval (CINTL) in this method is the fundamental time unit of this program; it is picked to give the desired accuracy in integration of continuous functions. Times involving discontinuities are calculated with much greater precision; pulse widths for the PWM and pulse delays are located with the full accuracy of the computer, the firing times of the PFMs are found to any specified precision.

The PFMs are double polarity devices, with the firing threshold the same for either polarity. The firing thresholds (RR(I)) may be set individually for each PFM. One firing tolerance (TOLER) is specified for all PFMs. The firing times are located with sufficient accuracy that firing occurs for PFM I when RR(I) $\leq$|integrator output of PFM|$\leq$ RR(I) + TOLER. For very rapidly changing functions this implies extreme precision in firing times, and as this may not be worthwhile past a certain point, the minimum size of the time step resorted to in this search (IMIN) is specified by the programmer. The time of each PFM firing, the PFM involved, and the polarity of its output may be printed out if desired. This is useful in detecting exact periodicities. (It is suggested that this print out be suppressed if large inputs occur for any nonsaturating PFM, as several hundred firings per second may occur in this situation.)

For the sampler-clampers all sampling intervals (LS(I)) must be integral multiples of CINTL, but all may be specified independently. All sampling occurs at the beginning of a CINTL, the one in which this first occurs (IS(I)) is specified for each sampler (0 $\leq$ IS(I) < LS(I)). No provision is made for random sampling.

The system may include up to five delayed functions. Each delay must be an integral multiple of CINTL. The function delayed may be any bounded quantity in the system, preferably a continuous quantity. The delayed function is evaluated at boundaries between CINTLs. Linear approximations are used for other times when needed in calculations. This is satisfactory provided CINTL is small or the delayed function is not rapidly changing.

Program LOVELL will handle one delay of the output impulse of a PFM. The length of this delay need not be an integral multiple of CINTL; hence short delays are readily handled. The length of the pulse delay is accurate to approximately ten decimal digits. More than one firing of this special PFM in any one CINTL will usually produce errors. The same procedure is used to handle the pulse width of a PWM, with the same accuracy.

Initial values may be set for all PFM integrators, all state variables of the continuous system, and all clamped quantities. The initial time is always zero.

The system output, consisting of any one combination of state variables, clamped quantities, time, and delayed functions, may be printed out at the end of each CINTL, or at any multiple of this time. A limit may be placed on output magnitude to terminate the problem in case of an unstable system. Changes to the program to print out multiple outputs may be easily made.

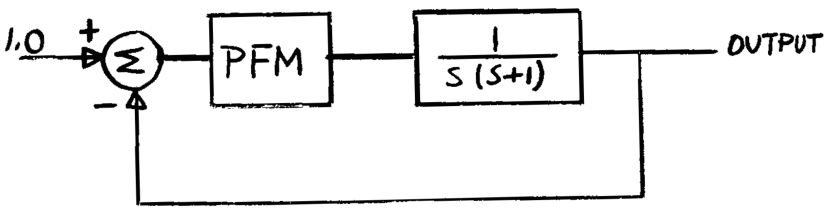
Problems may be stacked, to be run one after the other, resulting in a large saving of computer time. They may be related, but need not be.

All of the subroutines to be supplied by the user are to be written in 3600 Fortran.

INSTRUCTIONS-SUBROUTINES

The system equations for this program are supplied by the user through five subroutines, not all required for any one problem. The basic approach is illustrated below, and numerous examples with special complications follow later.

Simple PFM system

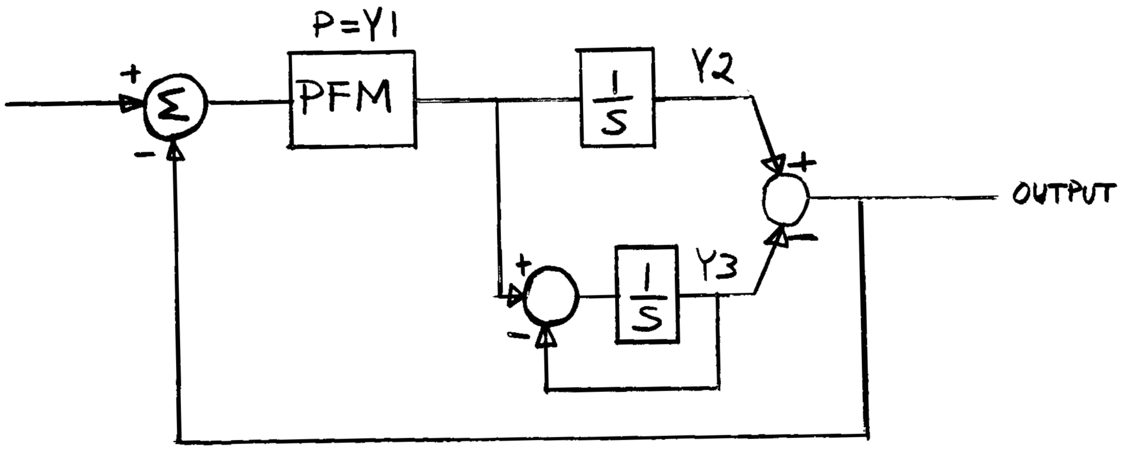


Let all initial conditions be zero. Let the PFM have a threshold level of 0.25. Let it produce impulses of area 0.50. Let the equation describing the PFM integrator between pulses be

$$\dot{p} = \text{input to PFM} - 0.40\ p$$

Calculation is desired over a period of five seconds, with print out of output every half second, and with print out of PFM firings. Use CINTL of 0.05 sec. Locate PFM firings to a tolerance of 0.0025, provided that time increments are not required smaller than 0.00001 sec. Stop if output exceeds 10.0

First it is necessary to write state variable equations for the system. Any set will do. Here a convenient procedure is to use partial fraction expansion of the plant, yielding



$$\dot{Y}(1) = 1.0 - 0.40\ Y(1) - Y(2) + Y(3) - 0.25\ \mathrm{SGN}(Y(1))\delta(|Y(1)| - 0.25)$$

$$\dot{Y}(2) = 0.0 + 0.50\ \mathrm{SGN}(Y(1))\ \delta(|Y(1)| - 0.25)$$

$$\dot{Y}(3) = -Y(3) + 0.50\ \mathrm{SGN}(Y(1))\delta(|Y(1)| - 0.25)$$

All PFM integrators are assigned state variable numbers before any plant or compensator state variable is numbered. State variable numbers must start with one and increase consecutively.

Program LOVELL handles the impulses separately, and these equations are now rewritten

$$\dot{Y}(1) = 1.0 - 0.40\ Y(1) - Y(2) + Y(3)$$

$$\dot{Y}(2) = 0.0$$

$$\dot{Y}(3) = -Y(3)$$

If the number of equations here is NEQ, relable the derivatives in order, as Y's, starting with Y(NEQ + 3). If time appears explicitly in these equations, it is replaced by Y(NEQ +1).

$$Y(6) = 1.0 - 0.40\ Y(1) - Y(2) + Y(3)$$

$$Y(7) = 0.0$$

$$Y(8) = -Y(3)$$

These equations are placed directly in subroutine DERFUN.

The PFMs are automatically reset to zero when they fire. Additional effects of the firing of PFMs are described by a weighting matrix of NEQ rows and NUM columns, where NUM is the number of PFMs. Note that a PFM may reset itself to other than zero if desired, by means of this matrix. In the present example

$$W = \begin{vmatrix} 0.0 \\ 0.5 \\ 0.5 \end{vmatrix}$$

This matrix is read in on problem data cards to be discussed later.

One may print as output any desired quantity in the problem, not necessarily the real output. This is defined in subroutine OUTPUT. For the problem considered this would have the statement OUT = Y(2) - Y(3)

The other subroutines would not be needed for this problem.

INSTRUCTIONS-DATA CARDS

Program LOVELL uses up to twelve types of data cards for each problem. Many fewer are used for simple problems. (In some cases of stacked problems only one is required. See Problem Stacking.) The READ statements and associated formats will all be given here, the symbols will be defined, and the set required by the illustrative example will be specified.

1
```
      READ 80, ID
   80 FORMAT(10A8)
```

Required for all problems. This may be any identification desired for the problem. The card may be blank, but it must be present.

2
```
      READ 81,NEQ,NUM,NS,NNS,JPNT,MPLM,NFD,IPD,IPWM,IDPFM,NSGN
   81 FORMAT(11I5)
```

Required for all problems.

NEQ is the number of PFMs plus the number of plant and compensation state variables. $0 < \text{NEQ} \leq 40$.

NUM is the number of PFMs. $0 \leq \text{NUM} \leq 10$. Blank is equivalent to 0.

NS is the number of sampler-clampers. $0 \leq \text{NS} \leq 10$. Blank is equivalent to 0.

NNS controls the printing of the output. Time and output will be printed at time 0.0 sec. and at the end of every NNS calculation intervals. If NNS is left blank printing will occur at the end of every calculation interval. The purpose of NNS is to allow the use of small calculation intervals without producing reams of data.

JPNT = 0 means print out for each firing of a PFM the time, the number of the PFM, and the sign of the output of the internal integrator of that PFM at firing. JPNT = 0 or blank suppresses this print out.

MPLM is the multiple problem indicator. Blank or 0 means no following problem; 1 means that another problem follows and a complete new set of data cards is to be read; 2 means that another problem follows, but only a new ID card is to be read as data. Any other changes will be handled by subroutine CHANGE. The last is very convenient when closely related problems are stacked.

NFD is the number of delayed functions. $0 \leq NFD \leq 5$.

IPD is the indicator for a pulse delay; 1 means pulse delay, blank or 0 means none.

IPWM is the indicator for a pulse width modulator; 1 means that there is one, blank or 0 means that there is not.

IDPFM is the indicator for discrete PFMs; 1 means that all PFMs are discrete, blank or 0 means that they are normal.

NSGN is the number of PFMs used as sign function generators.

3
```
      READ 83, CINTL, TMAX, ZMAX, TOLER, XMIN, PWMA, PWMB
   83 FORMAT(8F10.5)
```

Required for all problems.

CINTL is the basic time increment (in seconds) used in the Runge-Kutta solution.

TMAX is the problem time (in seconds) for which calculation is desired.

ZMAX is the cutoff value for output. If output magnitude exceeds ZMAX, calculations are terminated in that problem.

TOLER is the allowable difference between a PFM integrator and its threshold level when it fires.

XMIN is the minimum size of the time step resorted to in the search for firing times that will satisfy

threshold $\leq$ |integrator of PFM| $\leq$ threshold + TOLER.

If, because of XMIN, the search is stopped before this is satisfied, firing is said to occur with |integrator of PFM|> threshold + TOLER.

PWMA is the amplitude of the pulse emitted by the PWM. It may be left blank if there is no PWM.

PWMB is the saturation level for the input to the PWM, i.e., if input to PWM PWMB, the output pulses become continuous. It may be left blank if there is no PWM.

4
```
      READ 83, (RR(I),I = 1,NUM)
   83 FORMAT(8F10.5)
```

Used only if NUM is not equal to 0.

RR(I) is the threshold or firing magnitude for PFM unit I. RR(I) > 0.

Eight values are read per card; two cards are possible.

5
```
      READ 83,((W(I,J),I = 1, NEQ),J = 1,NUM)
   83 FORMAT (8F10.5)
```

Used only if NUM is not equal to 0.

Weighting matrix for the effect of PFM pulses on the Ys. W(I,J) is the weight used in calculating the effect of PFM J on state variable I. Entries may be 0.0. This matrix is read in by columns, eight entries per card. There may be up to 50 cards, but usually only one or two are needed.

6
```
      READ 83, (WD (I),I = 1,NEQ)
   83 FORMAT(8F10.5)
```

Used only if IPD is not equal to 0.

One column weighting matrix for a delayed pulse. There may be up to 40 rows, and hence up to 8 cards.

7
```
      READ 83, PD
   83 FORMAT(8F10.5)
```

Used only if IPD is not equal to 0.

PD is the length of the pulse delay, normalized with respect to CINTL. It need not be any integer, and must be expressed as a floating point number.

8
```
      READ 83,(Y(I),I = 1,NEQ)
   83 FORMAT(8F10.5)
```

Required for all problems.

Initial values of PFM integrators and other state variables of the continuous system, eight per card. There may be up to 5 cards.

9
```
      READ 81,(LS(I),I = 1,NS)
   81 FORMAT(11I5)
```

Used only if NS is not equal to 0.

LS(I) is the sampling interval for sampler-clamper I, normalized with respect to CINTL. It must be an integer.

10
```
      READ 81,(IS(I),I = 1,NS)
   81 FORMAT(11I5)
```

Used only if NS is not equal to 0.

IS(I) is the first sampling time of sampler-clamper I, normalized with respect to CINTL. It must be an integer. $0 \le IS(I) < LS(I)$.

```
11        READ 83, (CL(I),I  1,NS)
       83 FORMAT(8F10.5)
```

Used only if NS is not equal to 0.

Initial values of clamper outputs. There may be two cards.

```
12        READ 81,(LFD(I),I  1,NFD)
       81 FORMAT(11I5)
```

Used only if NFD is not equal to 0.

LFD(I) is the length of function delay I, normalized with respect to CINTL. It must be an integer.

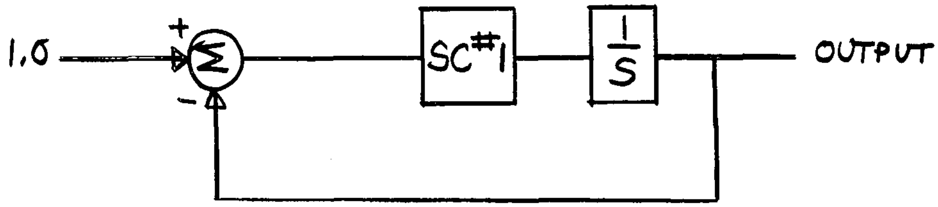
For the illustrative example previously specified the following data cards are required.

```
SIMPLE PFM SYSTEM

   3    1    0    10    1    b    b    b    b    b

 0.05       5.0       10.0       0.0025    0.00001       b        b

 0.25

 0.0        0.50       0.50

 0.0        0.0        0.0
```

SAMPLER-CLAMPERS

Program LOVELL will handle up to ten sampler-clampers. These operate at the beginning of specified computation intervals. The output of sampler-clamper I is denoted by CL(I). This quantity may be used in the definitions in subroutines DERFUN, OUTPUT, CLAMP, AND DELAY. The input to each sampler-clamper is defined in subroutine CLAMP, where the input to sampler I is denoted by CL(I + 10). Units used are numbered; the numbers must start with 1 and increase consecutively.

Sample problem



The sampler-clamper coperates every 0.3 seconds, starting with time 0.0. All initial conditions are 0. Use CINTL of 0.1 second. Compute to 6.0 seconds, printing output every 0.2 seconds. Stop if the output exceeds 10.0

Data cards

```
SIMPLE SAMPLER-CLAMPER PROBLEM
  1    0    1    2
0.1       6.0       10.0
0.0
  3
  0
0.0
```

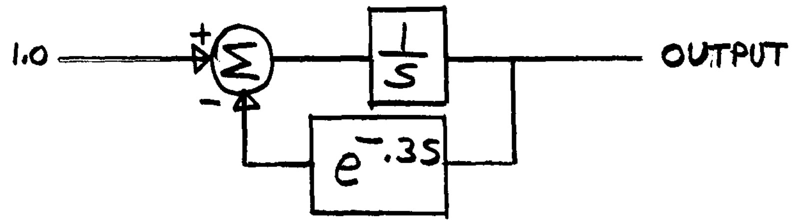
Subroutines

```
OUTPUT
  OUT = Y(1)
DERFUN
  Y(4) = CL(1)
CLAMP
  CL(11) = 1.0 - Y(1)
```

DELAYED FUNCTIONS

Program LOVELL will handle up to five delayed functions. The lengths of the delays must all be integral multiples of the calculation interval. The output of delay I is denoted by DELF(I). This quantity may be used in the definitions in subroutines DERFUN, OUTPUT, CLAMP, and DELAY.* The input to delay I is denoted by DD(I), and must be defined in subroutine DELAY. Units used are numbered; the numbers must start with 1 and increase consecutively.

Sample problem



All initial conditions are 0. CINTL = 0.1 second. Compute to 5 seconds, printing the output every 0.1 second. Stop if the output exceeds 3.0 in magnitude.

Data cards

```
SIMPLE CONTINUOUS SYSTEM WITH A DELAYED FUNCTION
 1   b   b   b   b   b   1   b   b
0.1     5.0     3.0
0.0
 3
```

Subroutines

```
OUTPUT
   OUT = Y(1)
DERFUN
   Y(4) = 1.0 - DELF(1)
DELAY
   DD(1) = Y(1)
```

----

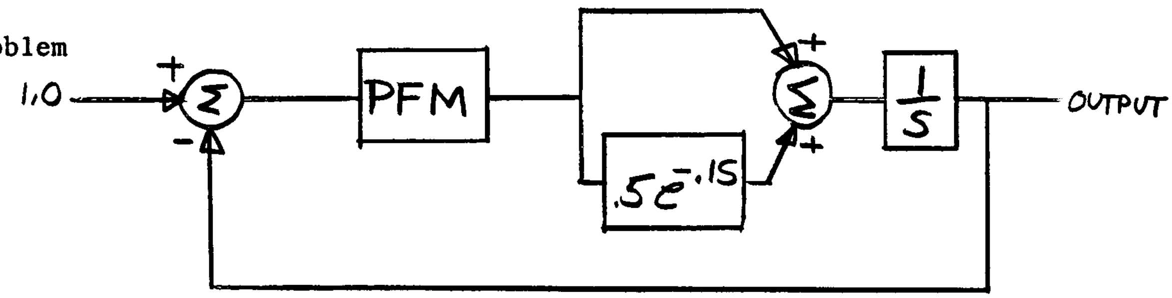
*One delay may directly feed another if desired.

DELAYED PULSES

Program LOVELL will handle one delay of the output impulse of a PFM unit. The delay may be of any size, but is normalized with respect to the calculation interval, and must be expressed as a decimal. It may be less than 1.0. The PFM unit which produces the impulses to be delayed must be the highest numbered one in the system. It may also produce a normal non-delayed output impulse. The effects of the delayed impulse are given by a weighting function WD(I), exactly like W but of only one column.

Restriction: For delays of 1.0 CINTL or greater, this special PFM must not fire more than once per CINTL. For delays of less than 1.0 CINTL the exact situation is quite complicated, but a conservative rule is that there should be no pulse currently delayed when this PFM fires.

Sample problem



Initial conditions are 0. The PFM produces a delayed output impulse of area 0.5 and a non-delayed output impulse of area 1.0. The threshold is 0.50. The PFM is a non-linear PFM with $\dot{p} = 1.0 - OUT - 2p^3$. Use CINTL = 0.2 seconds. Print output every second for 5 seconds.

Data cards

```
PFM WITH DELAYED AND NON-DELAYED OUTPUTS
  2    1    0    5    1    0    0    1
0.2       5.0       15.0      0.01       0.00001
0.5
0.0       1.0
0.0       0.5
0.5
0.0       0.0
```

Subroutines

OUTPUT

```
OUT = Y(2)
```

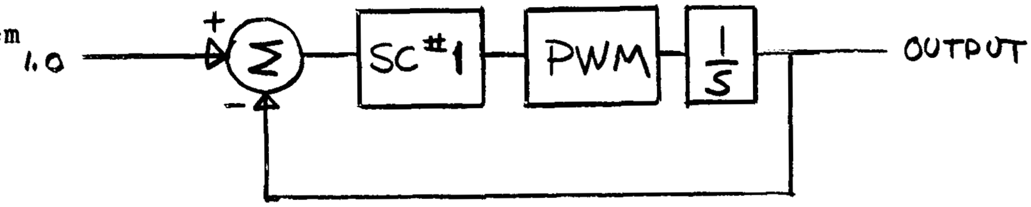
DERFUN

```
Y(5) = 1.0 - Y(2) - 2.0*Y(1)**3 $ Y(6) = 0.0
```

PULSE WIDTH MODULATION

Program LOVELL will handle one PWM in place of a delayed impulse. The PWM is always considered to follow sampler-clamper 1 in order to set the timing for the modulator. There may be other sampler-clampers and/or PFMs in the circuit. The output of the pulse width modulator is denoted by PWM. This may be used in subroutines DERFUN, OUTPUT, CLAMP, or DELAY, but is normally used in DERFUN. The magnitude of the output pulses is PWMA. The saturation level of the input, i.e., the clamper output which will produce a continuous PWM output is PWMB.

Sample problem



Initial conditions are 0. The sampler-clamper should operate every 0.3 seconds starting with time 0.0. The magnitude of the PWM output is 1.0; the input saturation level is 0.5. Use CINTL 0.1 second; print the output every 0.4 seconds for a period of 8 seconds.

Data cards

```
SIMPLE PWM SYSTEM
  1    0    1    4    1    0    0    1
0.1       8.0       20.0       b       b    1.0    0.5
0.0
  3
  0
0.0
```

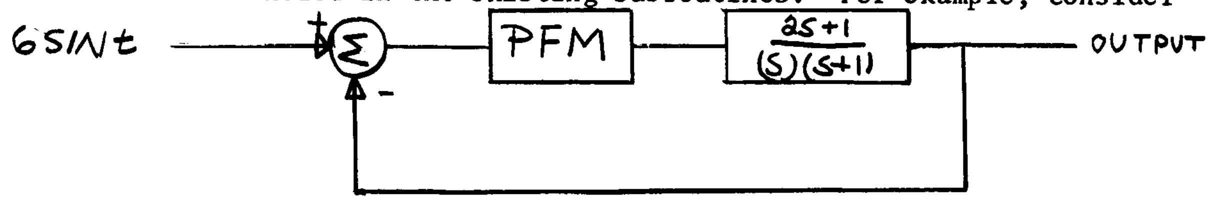
Subroutines

```
OUTPUT
  OUT = Y(1)
```

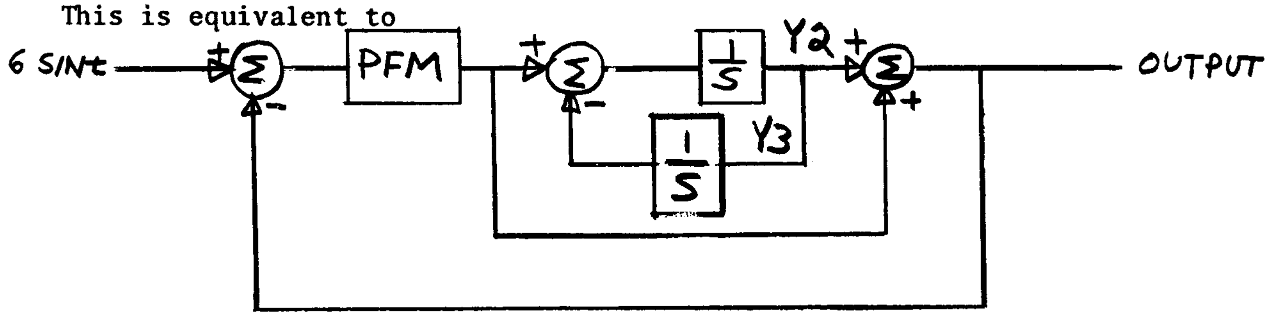
```
DERFUN

   Y(4) = PWM

CLAMP

   Y(11) = 1.0 - Y(1)
```

SATURATION

Saturation is best handled in the existing subroutines. For example, consider



This is equivalent to



The PFM is a NPFM, described by $\dot{p}$ = input to PFM - p

Normally subroutine DERFUN would contain

```
Y(6) = 6.0*SINF(Y(4)) - Y(2) - Y(3) - Y(1) $ Y(7) = -Y(2) $ Y(8) = 0.0
```

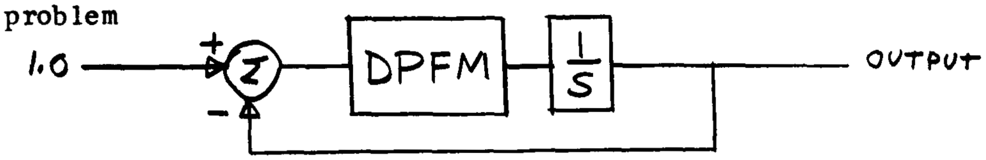
If the input to the PFM saturates at a level of 2.5, DERFUN is changed to

```
AR = 6.0*SINF(Y(4)) - Y(2) - Y(3)

IF (ABSF(AR).GT.2.5)AR = SIGNF(2.5,AR)

Y(6) = AR - Y(1) $ Y(7) = -Y(2) $ Y(8) = 0.0
```

DISCRETE PULSE FREQUENCY MODULATORS

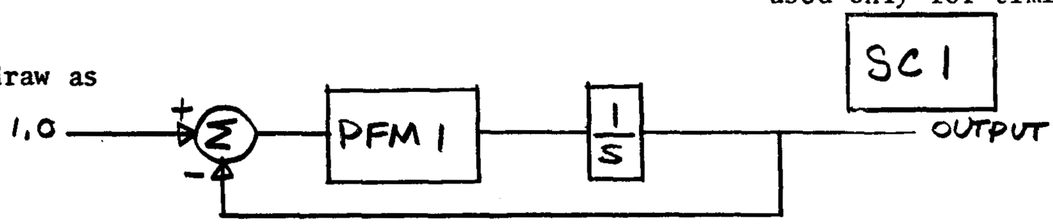
The discrete pulse frequency modulator (DPFM) can only fire at specific times. It will fire at these times if the output of its internal integrator exceeds the threshold value. Program LOVELL will handle DPFM systems, but in any one system all PFMs must be discrete or none may be. If the PFMs are discrete a sampler-clamper must be associated with each one for timing. The identifying number of the sampler-clamper must be the same as that of the associated PFM. (Note that if a PWM is also present it uses SC 1; hence in that case PFM 1 is a dummy unit.) For these modulators TOLER must be set so large as to be inoperative, e.g., 1000.

Sample problem



used only for timing

Redraw as



Initial conditions are 0. The DPFM may fire at times 0.2k seconds, where k is an integer. The DPFM emits unit impulses and is of NPFM type with $\mathring{p}$ = input to PFM - p. Threshold level is 0.25. Take CINTL 0.1, print out every 0.1 seconds for 10 seconds.

Data cards

```
SIMPLE DPFM SYSTEM
  2   1   1   1   1  0   0   0   0   1
0.1      10.0      20.0     1000.0  0.001
0.25
0.0       1.0
0.0       0.0
  2
  0
0.0
```

Subroutines

OUTPUT

```
OUT = Y(2)
```

DERFUN

```
Y(5) = 1.0 - Y)1) - Y(2) $ Y(6) = 0.0
```

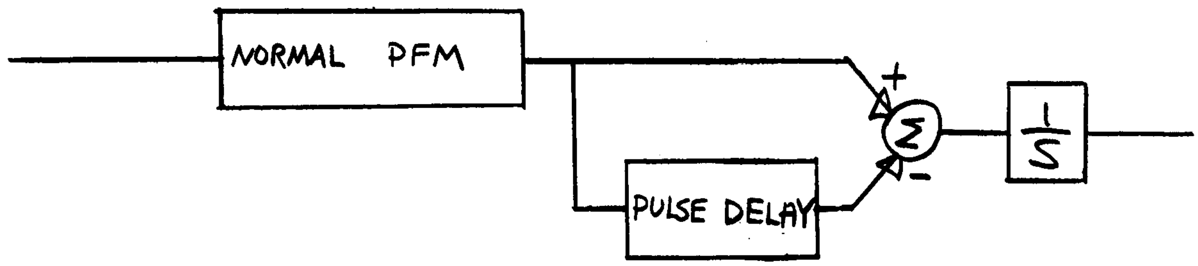
CLAMP

```
CL(11) = 0.0
```

PFM WITH FINITE PULSE WIDTH OUTPUT

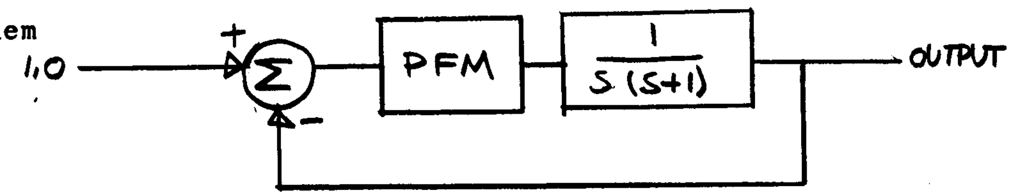
If desired an output pulse with a finite width may be produced by one PFM. This uses the pulse delay mechanism, and hence the problem cannot also include a delayed pulse or a PWM. This PFM must not fire while still producing a pulse from a previous firing (errors will result); if there is doubt concerning this, the PFM can be designed with a saturating input. This type of modulator allows a check to be made of the effects of a real rather than ideal PFM.

The equivalent circuit for the finite pulse width PFM is

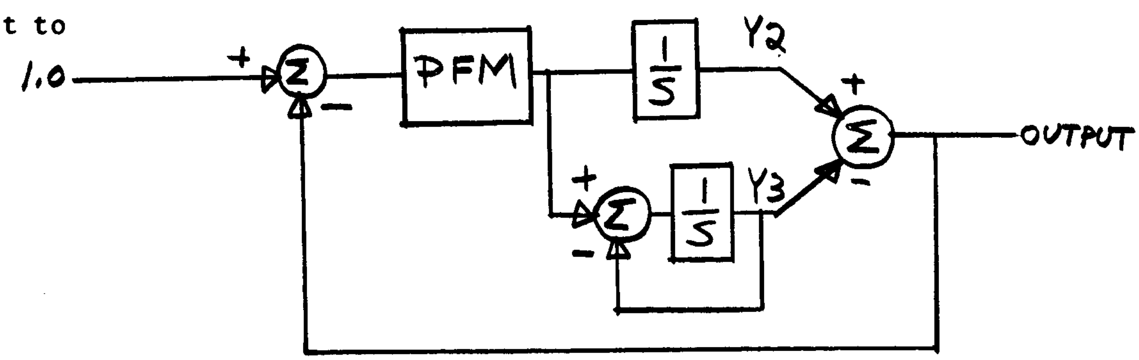


The weighting matrices are used to produce the proper pulse height, and the pulse delay determines its width.

Sample problem



equivalent to



All initial conditions are 0. The PFM is described by $\dot{p}$ = input to PFM - p. Threshold is 0.25. Use CINTL = 0.05, print output every 0.2 seconds for 5 seconds. Stop if output exceeds 20.0. The PFM output has an area of 0.25. If this is an impulse output, the data cards and subroutines would be as follows.

```
NORMAL PFM

  3    1    0    4    1    0    0    0    0    0

0.05        5.0        20.0      0.002      0.00001

0.25

0.0         0.25       0.25

0.0         0.0        0.0

OUTPUT

   OUT = Y(2) - Y(3)

DERFUN

   Y(6) = 1.0 - Y(1) - Y(2) + Y(3) $ Y(7) = 0.0 $ Y(8) = -Y(3)
```

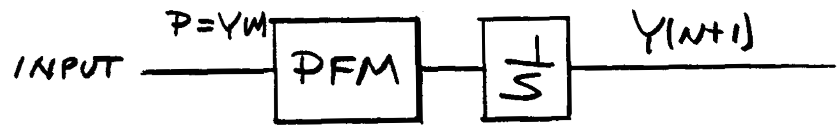
If the pulses still have area 0.25, but are 0.01 second long and 25 units high, the data cards and subroutines would be as shown. The output of the separate integrator in the equivalent circuit is labelled Y(4).

```
REAL PFM

  4    1    0    4    1    0    0    1    0    0

0.05        5.0        20.0      0.002      0.00001

0.25

0.0         0.0        0.0       25.0

0.0         0.0        0.0      -25.0

0.2

0.0         0.0        0.0        0.0

OUTPUT

   OUT = Y(2) - Y(3)

   DERFUN

      Y(7) = 1.0 - Y(1) - Y(2) + Y(3) $ Y(8) = Y(4) $ Y(9) = Y(4) - Y(3)

      Y(10) = 0.0
```

SGN FUNCTION GENERATOR, BANG*BANG, IDEAL CONTACTOR WITHOUT DEADBAND

Whatever title one chooses there is sometimes need for a box whose output is 1.0 if the input is greater than 0 and -1.0 if the input is less than 0. This can be produced by suitably misinstructing a normal PFM. The output changes of this device are not restricted to occur at the end of calculation intervals, but can be located as accurately as desired.

The equivalent circuit for the device is

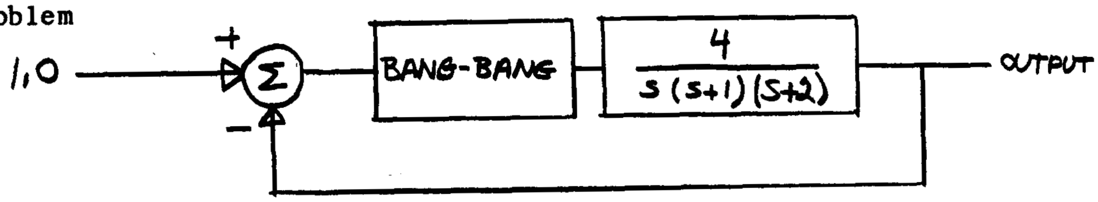


The integrator of the PFM is described by $\dot{p}$ = derivative of INPUT. The threshold is picked so large as not to be reached in normal operation. For initial conditions,

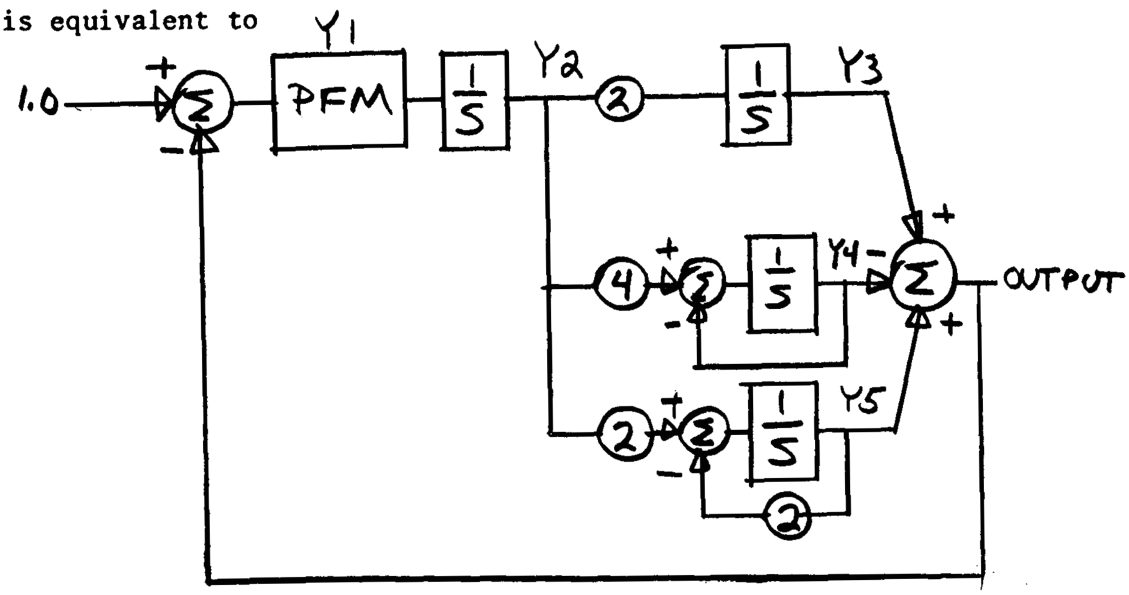
Y(n) = -(threshold)(SGN of initial input to device) + (initial input to device)

Y(n + 1) = SGN of initial input to device.

Sample problem



This is equivalent to



Use CINTL = 0.1 and TOLER = 0.001. Print the output every 0.5 seconds for 5 seconds. All initial conditions are 0. Use a threshold of 300.

Data cards

```
SIMPLE BANG-BANG SYSTEM
  5    1    0    5    1    0    0    0    0    0    1
 0.1        5.0        20.0       0.001     0.00001
 300.0
-600.0      2.0         0.0       0.0        0.0
-299.0      1.0         0.0       0.0        0.0
```

Subroutines

```
OUTPUT
     OUT = Y(3) - Y(4) + Y(5)
DERFUN
     Y(8) = 2.0*Y(5) - Y(4) $ Y(9) = 0.0 $ Y(10) = 2.0*Y(2)
     Y(11) = 4.0*Y(2) - Y(4) $ Y(12) = 2.0*Y(2) -2.0*Y(5)
```

Note particularly that when the PFM unit used as a SGN generator fires it is not reset to zero, but nearly to the opposite threshold.

This type of unit MAY NOT BE USED with discrete PFM systems.

The PFMs used in this way must be the low numbered ones, starting with 1. The number of PFMs so used must be given on data card 2.

PROBLEM STACKING

It takes about 35 seconds to compile this program; simple systems can then be handled in 1 to 3 seconds each. Hence problem stacking is advised whenever more than one problem is to be solved, whether or not the problems are in any way related.

The multiple problem symbol is MPLM, read in on data card 2.

If MPLM is 0 or blank, the current problem is the last (or only) one.

If MPLM is 1, the current problem will be followed by another, assumed to be completely unrelated. A complete set of data cards will be read for this following problem.

If MPLM is 2, the current problem will be followed by another, assumed to be more or less related to the present one. Only a new ID card will be read. Subroutine CHANGE will be called to read in any new parameters desired as well as to reset initial conditions (either to the same or to different values). It is suggested that sufficient information be included on the ID card to identify the changes made.

The major technique used in handling stacked problems is the subroutines and the problem counter M2. The latter is 1 for the first problem of a stack and is incremented by 1 for each succeeding problem. It is used as follows.

Suppose that among other differences in 4 stacked problems the desired output for problems 1 and 3 is Y(1); for problem 2 it is 3.2 - Y(7); and for problem 4 it is -Y(1) - Y(2). Subroutine OUTPUT, in addition to the cards supplied with the program, would consist of,

```
  GO TO (1,2,1,3),M2
1 OUT = Y(1) $ RETURN
2 OUT = 3.2 - Y(7) $ RETURN
3 OUT = -Y(1) - Y(2)
```

Similar treatments are used in DERFUN, CLAMP, and DELAY. A little thought will often save many cards. For example, a sequence of related problems with only an incremented input might have for DERFUN

Y(6) = -Y(1) + 4.4 + M2*0.1 $ Y(7) = Y(3) $ Y(8) = -Y(3)

Subroutine CHANGE is used when groups of consecutive problems are sufficiently related that many parameters do not need to be changed. There may be more than one such group in a stack. Subroutine CHANGE includes data for the initial values of Y(I) and CL(I), up to the number of each used in the problems. These may be identical or may vary with M2. In addition, if it is desired to change any parameter whatsoever which is normally supplied on data cards, the new values are simply read in when the desired value of M2 occurs.

It is also necessary in subroutine CHANGE to remove the "repeat" signal at the end of a group of related problems. If the problem calling CHANGE is the last of the stack, set MPLM to 0; if an unrelated problem follows, set MPLM to 1. For example, if the stack consists of a group of 4 problems followed by a different group of 3 problems, data card 2 of problems 1 and 5 (the only problems for which a complete set of data cards are read) would contain MPIM. 2. CHANGE would include the statements (or their equivalents) IF(M2.EQ.4)MPIM = 1 $ IF(M2.EQ.7) = 0.

ACCURACY

The following is a very useful rule of thumb. When integrating quantities describable by a complex frequency s, the maximum error in the integral over one calculation interval will be about 0.002% if $|s| \cdot \text{CINTL} \leq 0.5$. Halving the CINTL will reduce the error by a factor of 16. These errors are not random, and when integrating an oscillatory function they tend to cancel.

NON-ZERO STARTING TIME

It is sometimes desired to start (or restart) problems at a time other than zero. While this can be accomplished by rewriting the equations, the print out will contain the incorrect time. Hence provision has been made for entering an initial time on data card 3. The READ instruction is now

```
      READ 83,CINTL,TMIN,TMAX,ZMAX,TOLER,XMIN,PWMA,PWMB
   83 FORMAT (8F10.5)
```

The new quantity is TMIN, the initial time in seconds. This must be specified for all problems. As with all parameters it may be varied by subroutine CHANGE.