ANALAG Functions for Digital Filtering of Numeric Signals

ANALAG S37

DESCRIPTION

The ANALAG function provides a digital filter for smoothing fluctuating numeric values, such as those resulting from process control measured variables when fed into a GEM 80 controller via analogue input modules or blocks.

It provides exponential (first order) smoothing of input changes.

The mnemonic ANALAG stands for ANAlogue LAG.

FEATURES

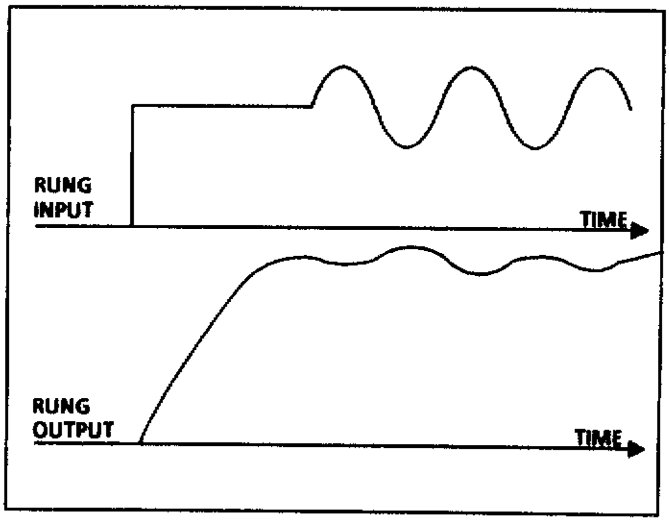
- Equivalent to a single time constant filter in analogue systems
- User programmable time constant
- Provision for switching the filtering action into or out of the system, and for initialising the output

SPECIFICATION

Control language:-

```
        ANALAG
X-------SPEC.-------VALUE-------Q
         S37
```

Number of value locations	:	5
Equation when filter switched out	:	$Q = X$
Equivalent equation when filter switched in	:	$Q = KX + (1-K)Q'$ where Q' = output value on previous scan



When filter switched in:

1. If input X remains constant, Q will eventually reach a steady level (i.e. $Q = Q'$) equal to X.

2. K can only be set as a multiple of 0.001. The smaller the value of K, the smaller the influence of the input X on the output Q and hence the greater the smoothing.

3. The equivalent time constant provided by ANALAG is approximately the repetition interval divided by K. For errors in this approximation, see Examples below.

4. The internal calculations within ANALAG often result in a non-integer value for the desired output. Since Q can only be an integer, any fractional part which cannot be output is held over and used in the calculation for the next value of Q. The exact equation used for evaluating Q is:

$$Q = Q' + \frac{G(X - Q') + R}{1000}$$

where $G = 1000K$ and $R = 1000$ times the remainder from previous calculation of Q.

APPLICATION NOTES

EXAMPLES OF ANALAG RESPONSE

1. **Step Response**

Suppose that K = 0.500 (a short time constant).

The equivalent equation for Q is:

$$Q = 0.5X + (1 - 0.5)\,Q'$$

$$= 0.5X + 0.5Q'$$

Suppose that, initially, Q = Q' = X = 0, and that X is then switched from 0 to 32. From the equivalent equation, since 0.5X = 16, the value of Q will be given by:

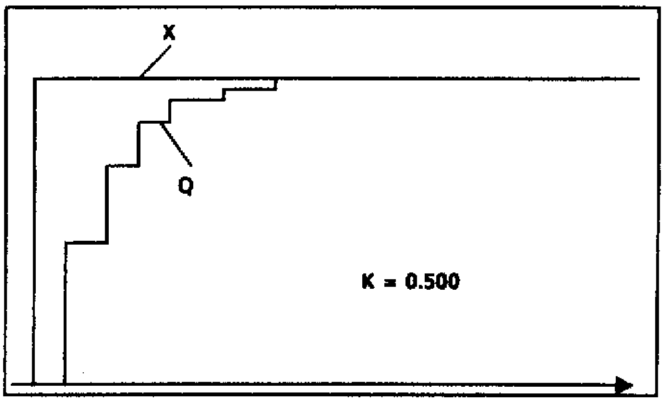
$$Q = 16 + 0.5Q'$$

and Q will take on values on successive scans as follows:

16	(= 16 + 0)
24	(= 16 + 0.5 x 16)
28	(= 16 + 0.5 x 24)
30	(= 16 + 0.5 x 28)
31	(= 16 + 0.5 x 30)
31	(= 16 + 0.5 x 31 to the nearest integer)
32	(= 16 + 0.5 x 31 + remainder)
32	(= 16 + 0.5 x 32)

and the output will remain at 32 thereafter till X changes again.

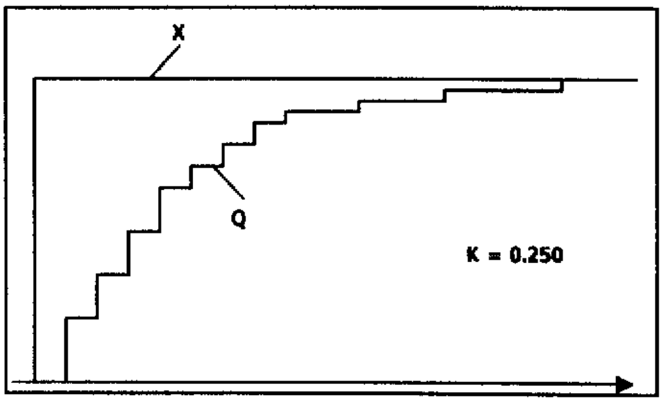
As a graph, the output Q is as follows, where the time scale is drawn in units of the program repetition interval:



Suppose the example is repeated for the same input conditions but with K = 0.250 (i.e. roughly double the time constant. A similar calculation to that above will show that Q takes on the values on successive scans of:

8, 14, 18, 22, 24, 26, 28, 29, 29,
30, 30, 31, 31, 31, 31, 32 ....

giving a graph of:



Note that both these responses look rather 'lumpy' since, in order to keep the lists of figures short, the examples chosen have very short time constants and the input step size is very small, only about 0.1% of full scale. It will be appreciated that, with lower values of K to give more smoothing, and with larger step inputs, response graphs look appreciably smoother.

2. Fluctuating Signal

Suppose that K = 0.250. The equivalent equation for Q is:

$$Q = 0.25X + 0.75Q'$$

Suppose that the input X is nominally 32 but with noise such that it cycles through values as follows:

16, 44, 48, 20, 16, 44, 48, 20, 16 etc.

The output Q will then cycle through the following corresponding values:

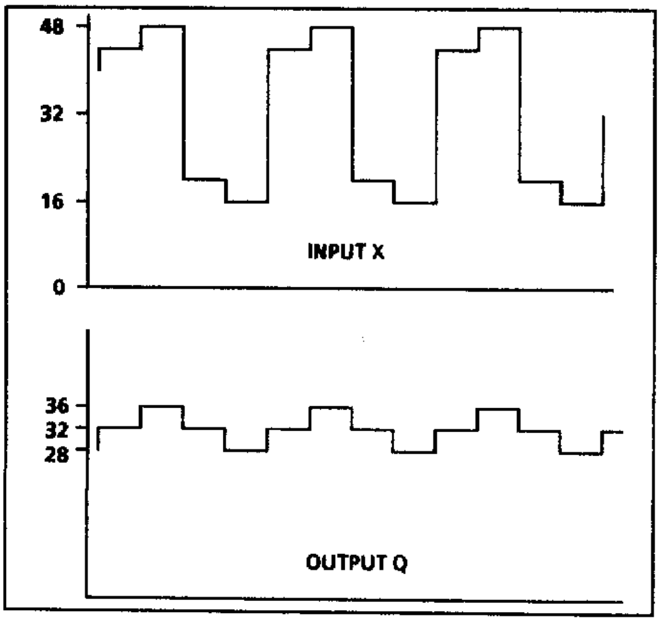
28, 32, 36, 32, 28, 32, 36, 32, 28 etc.

which come from:

28	= 0.25 x 16 + 0.75 x 32
32	= 0.25 x 44 + 0.75 x 28
36	= 0.25 x 48 + 0.75 x 32
32	= 0.25 x 20 + 0.75 x 36
28	= 0.25 x 16 + 0.75 x 32

etc.

Thus, whereas the input signal X varies ±16 either side of the mean of 32, the output Q only varies ±4, and the graphs of the input and output are as follows:



Note: The sequence of values used in this example have been chosen to make the calculations easy by not involving any remainders. In practice, it is highly probable that each calculation of Q will involve a remainder, which will be carried over to the calculation of the next value of Q.

The reduction in ripple on the output signal Q compared with the input signal X depends on (a) the value of the smoothing constant K, and (b) the frequency of the ripple. For ripple frequencies which are low compared with the program repetition frequency, and for low values of K, the attentuation is given by:

$$A = \frac{2\pi fT}{K}$$

Where f = noise ripple frequency (Hz)
T = program repetition interval
K = ANALAG smoothing factor

For the example, this formula cannot be expected to give an accurate answer since K is not low and the ripple frequency is only a quarter of the program repetition frequency. However, it would give a result as follows:

$$f = \frac{1}{4T} \qquad K = 0.250$$

$$\text{Hence } A = \frac{2\pi \times \frac{1}{4T} \times T}{0.250} = 6.28 \text{ times attenuation}$$

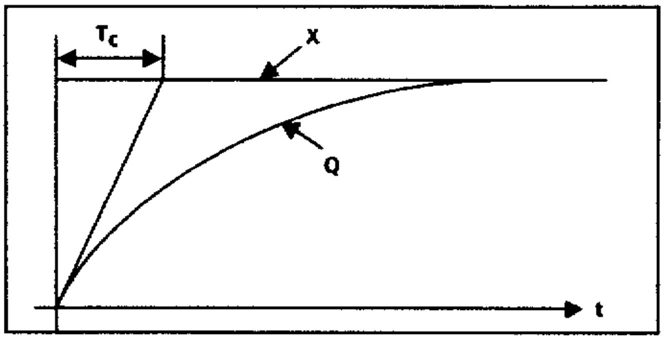
whereas, looking at the figures for X and Q, the excursion has only been attenuated about 4 times.

3. Time Constant

In assessing the performance of an analogue filter it is normal practice to work in terms of its time constant. The step response of a single time constant analogue filter, in terms of its time constant T_c, is given by:

$$Q = X(1 - e^{-t/T_c})$$

and the time constant T_c represents where a tangent to the response meets X, as below:



The step response of ANALAG is similar, as given earlier, and can also be treated in terms of a time constant.

Similarly, the attenuation of an analogue filter for a sinewave input is given by:

$$A = \omega T_c$$

where ω is the frequency of the sinewave in radians per second, for ω's high compared with $1/T_c$.

Further, the phase shift introduces between the attenuated output sinewave and the input sinewave (an important consideration in closed loop control systems) is given by:

$$\varnothing = \arctan(\omega T_c)$$

To set up ANALAG to give the equivalent of an analogue filter with time constant T_c, the smoothing factor K should be calculated from:

$$K = 1 - e^{-T/T_c}$$

where T = program repetition frequency
T_c = desired time constant

For time constants T_c which are large compared with the program repetition interval T, this equation reduces to:

$$K = T/T_c$$

Example values:

Program Repetition Interval T	Time Constant T_c	ANALAG Smoothing Factor K		
		Approx. (T/T_c)	Accurate ($1-e^{-T/T_c}$)	Accurate to Nearest 0.001
100ms	500ms	0.200	0.181	0.181
100ms	1s	0.100	0.095	0.095
100ms	5s	0.020	0.0198	0.020
100ms	10s	0.010	0.00995	0.010
100ms	50s	0.002	0.001998	0.002

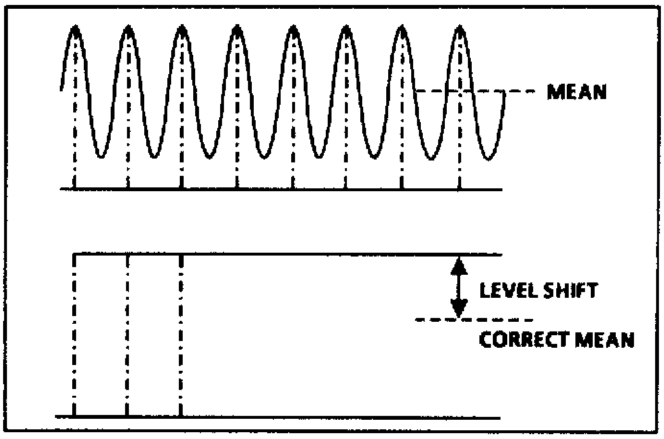
4. High frequencies

ANALAG and similar types of digital filter are not suitable for attenuating ripple of frequencies higher than (1/2) program repetition frequency.

For this reason, all GEM 80 analogue input units include hardware filtering for attenuating high frequencies.

The inability of digital filters to deal with high frequencies arises from the fact that, when such a signal is sampled, the ripple on the resultant digital signal is of a different, lower frequency. The effect is known as aliasing. If the alias frequency (in rad/s) is below $1/T_c$, where T_c is the time constant produced by ANALAG, then it will not be attenuated.

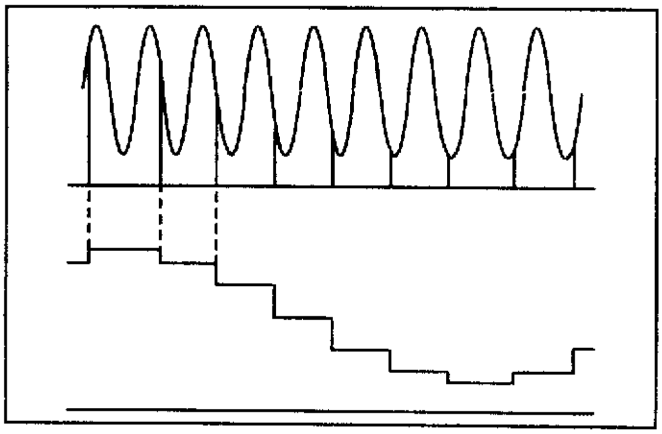
Consider a signal which contains ripple of a frequency precisely equal to the program repetition frequency. The digital signal obtained by sampling it will then be as shown below:



where it will be seen that the sampled signal has a shift of level compared with the mean of the input signal. A digital filter such as ANALAG cannot remove this level shift, as the signal into it appears exactly the same as a steady valued signal.

Similar effects occur with sampling if the signal carries ripple which is an exact multiple of the program repetition frequency.

If the ripple frequency is close to, but not equal to, a multiple of the program repetition frequency, the effect of sampling is to produce a signal containing ripple at the beat frequency between the two frequencies, e.g.



and the resulting frequency could be too low for ANALAG or any similar type of digital filter to attenuate. Hence the need for hardware filtering of high frequency ripple.

HOW TO PROGRAM ANALAG

Choose suitable write-enabled data table usable by special fuctions. Denoting the table address keyed below VALUE in the rung by Zm, then the 5 table locations required, the input X and the output Q are as given in Table 1.

DATA SET BY USER

Zm + 1 sets the value of K. Determine this from the required time constant, as given earlier, knowing the program repetition interval. Note that, if the program repetition interval is altered during programming the rest of the user program, then K must be re-calculated, and Zm + 1 reset.

Zm + 4 must be set to some non-zero value for the filtering to be switching in. Generally, this will be set to a constant (e.g. -1), but for some applications it may be changed under program control to zero when it is desired to switch the filtering action out.

Zm + 2 does not normally need to be set by the user. It may optionally be set by the user to an initial value, such that ANALAG calculates its output as though its previous output was not zero. This facility may be useful on changeover, e.g. from manual to auto, to achieve bumpless transfer.

Zm and Zm + 3 need no user set data.

CHOICE OF DATA TABLE

The following factors determine the choice of which table to use for the values Zm to Zm + 4:

1. Table must not be write-protected.

   If it were, ANALAG would not be able to store the values of Q' and R required for the next calculation on the next scan.

2. Whether Table contents are cleared on power up.

   User may want the values of Q' and R to be retained while the mains supply to the controller is off for some applications, whereas, for other applications, this may be unnecessary.

3. Whether Table contents are accessible by video. (This applies to controllers which include video only).

   If any of the values Zm to Zm+4 need to be displayed (numerically or graphically) in a video format, the table used must be accessible to video.

The user should refer to the Product Data Sheet for his particular controller to find the attributes of the various tables included in it.

EXAMPLE

(a) Requirements

Condition analogue input C0 with a 5s time constant filter. Controller has 100ms program repetition.

(b) Program rungs

```
|                                     |
+--<AND>-----------------------<OUT>--+
|   20                          G101  |
|                                     |
|                                     |
+------------------------------<OUT>--+
|                               G104  |
|                                     |
|         ANALAG                      |
+--<AND>---<SPEC>----<VALUE>----<OUT>--+
|   CO      S37        G100       D32  |
```

(c) Value data

G100	Fault code and flags
G101	Smoothing constant (set to 20 by first rung)
G102	Previous output
G103	Previous remainder
G104	Mode control (set to -1 by second rung)

(d) Remarks

This example illustrates the use of program rungs to set up 'value' data. For controllers which include functions LOCATE (S20) and MOVE (T20), the same effect may be obtained by entering the data in the preset table and copying to a working value table on start up. The MOVE function data sheet shows a typical program using the P and W tables for this purpose.

Table 1 Program rung and value data

Sym	Location	Use	Range
F	Zm	Fault code and flag set by ANALAG	See table 2
K	Zm+1	Smoothing Constant. Set by user.	1 to 1000 corresponding to values of K from 0.001 to 1.000 respectively
Q	Zm+2	Previous output, stored by ANALAG. Used in next scan.	±32767 (can be user initialised if required)
R	Zm+3	Previous remainder, stored by ANALAG. Used in next scan.	
M	Zm+4	Filtering in/out switch. Set by user.	0: Filtering out, Q = X Any non-zero value: Filtering in.
X	-	Input to ANALAG from rung	±32767
Q	-	Output from ANALAG	±32767

Table 2 Fault conditions and flags

Condition	Result	Code in Zm Value	Code in Zm Bits set
Programming Errors			
No VALUE after SPEC.	Q = 0	-	-
VALUE given number instead of address	Q = 0	-	-
VALUE address Zm in write-protected memory	Q = 0	-	-
VALUE address Zm in table not usable by special functions	Q = 0	-	-
Fifth address Zm + 4 beyond end of data tables memory area	Q = 0	-	-
Data Errors			
Zm + 1 contains number <1 or >1000	Q=0	5	Zm.0=1 Zm.2=1
Status			
Zm + 4 zero (filtering switched off)	Q = X Q' = X R = 0	9	Zm.0=1 Zm.3=1

CEGELEC
INDUSTRIAL CONTROLS

Kidsgrove, Stoke-on-Trent, ST7 1TW, England
Tel: (0782) 783511 Fax: (0782) 776329 Telex: 36293/4 CICKID G
Registered in England No. 2259095