CLOSED LOOP CONTROL SYSTEMS

USING GEM 80 PROGRAMMABLE CONTROLLERS

(1987 Edition)

CONTENTS

T451 Issue 1 04.87

GEC Industrial Controls Ltd.,
Kidsgrove, Stoke-on-Trent, Staffordshire ST7 1TW, England
Telephone: Kidsgrove (078 16) 3511
International 44 78 16 3511)
Fax (Group 3): Stoke-on-Trent (0782) 776 329
Telex: 36293 GECICK G

Holding Company : The General Electric Company p.l.c. of England,
1 Stanhope Gate, London W1A 2EH, England

PURPOSE OF THIS MANUAL

This manual is an introduction to control engineering using GEM 80 programmable controllers. It is intended for all engineers - those who have never done control engineering at college, those who did but have forgotten it, and for those who are expert on the theory but not so hot on the practice. As for those who know it all already, we think they may find the approach adopted useful for educating their juniors.

As an introduction to control engineering, the chief aim of this manual is to take the mystery out of a subject which is often shrouded in a mist of mathematical theory. Mathematics has been kept to an absolute minimum; engineers who have majored in control engineering may be appalled at the lack of mathematical treatment. There are no transfer functions, no Laplace transforms, no Nyquist diagrams, etc. If you want to learn something about such topics (and we hope you will want to), then there are plenty of textbooks available from which you can teach yourself the theory.

This manual is something different. It tries to keep clearly in sight what the reasons are for doing things in certain ways. It deals with what to look for when choosing measurement transducers, signal conditioning equipment, etc. It deals with how performance can be affected by where you place transducers on the plant. And so on. In other words, it teaches you about some of the practicalities which they don't teach you, or only skim over, at college or in textbooks.

LIMITATIONS

We have obviously had to write this manual in general terms, and not specifically for any particular industry or application. It will not necessarily tell you what is the best way of designing a system for your particular application. However, we hope it will direct you into "systems thinking" so that, when you start on your design, you will ask people (including yourself) the right sorts of questions.

KNOWLEDGE ASSUMED

1. We assume that you have some familiarity with programming GEM 80 controllers, including how to use the pre-written software routines called Special Functions.

2. We assume that you have no prior knowledge of control engineering.

3. Finally, we assume no mathematical ability, other than the ability to perform the standard four functions of addition, subtraction, multiplication and division. You won't even find an exponential in the text.

A NOTE ABOUT THE 1987 EDITION

This edition is our first attempt at producing a practical introductory manual on the design of closed loop control. It is already out of date, as newer models of GEM 80 Programmable Controller contain enhanced sets of Special Functions, some of which are not covered in this edition.

In the next edition, we will cover the use of some of these extra Special Functions. However, the other way in which the next edition will be an improvement on this one is that it will incorporate the results of any feedback we get from you.

When you have five minutes to spare, please use the sheet at the back of this manual to jot down any comments you have on this edition. Please tell us about items which you felt ought to have been included, were not adequately explained, or which you couldn't find in the Index. And, of course, tell us about any statements we have made which you don't agree with.

----o0o----

SECTION 1

ANALOG CONTROL

CONTENTS

(continued)

----oOo----

INTRODUCTION

As soon as we begin, we run up against our first bit of jargon - "open loop". It's a phrase used to mean a control which hasn't got any loop; it might better be called "no loop" control. All we do is take our setpoint, use it to drive some output signal, and hope that we get the right result. We don't use any measurement to check that the result is right, so we haven't got any feedback, so we can't have any loop.

For instance, suppose you want to control the flow of some fluid. As the control device, suppose you have a valve in the pipe, actuated by a 4 to 20mA signal. In open loop control, all we do is to take the setpoint and use it to drive a mA signal from a GEM 80 analog output module to the valve. We obviously need to add in an offset, such that when the setpoint is zero we get a 4mA output; we also need to multiply the setpoint by a suitable factor so that maximum setpoint gives 20mA output. In simple block diagram terms, our system looks like this:-

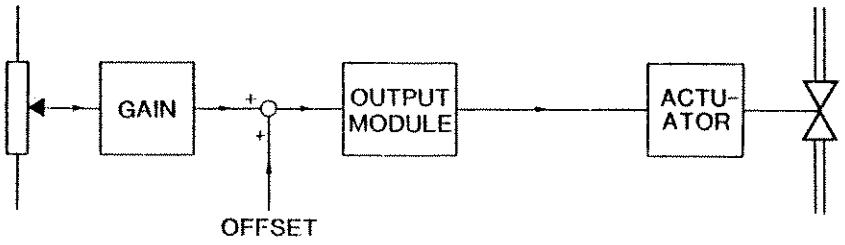

Fig. 1
Open Loop Control - Block Diagram

(Note: We have shown the setpoint diagrammatically as coming from a potentiometer. In a GEM 80 system, it is more likely to come from a keyboard, thumbwheel switch, or a separate supervisory GEM 80 controller. However, because the symbol for a potentiometer is easily understood as meaning a source of an adjustable signal, we will continue to use this symbol in all further block diagrams in this Section.)

In a GEM 80 ladder diagram, a gain with an offset can be provided by the Special Function LINCON, S11, so that a suitable rung would be:-

```
! SETPT  LINCON                                                ACTR !
+-<AND>---SPEC.---VALUE-------------------------------------<OUT>+
!  A5      S11      W200                                      B7  !
```

Fig. 2
Open Loop Control - GEM 80 rung

ANALOG CONTROL

Most users of GEM 80 quickly become expert at programming ladder diagrams for the sequence control of quite complicated sequences of operations. After all, designing a ladder diagram program is very similar to designing relay circuits, and you can make use of similar thought processes in the design of each.

However, when you come to analog control, and if you have not designed such controls before, you may feel a little unsure of yourself. Before the advent of advanced programmable controllers like GEM 80, the sequencing controls and the analog controls might well have been designed by different engineers, using entirely separate equipment (and their own jargon as well!). Now, with GEM 80, both types of control can be carried out in a single, integrated range of equipment.

There are plenty of textbooks available on closed loop control systems. However, many of these tend to be rather mathematical in treatment, and show you how to analyse a system - once you've decided what it is going to consist of! This Section is therefore intended to get down to basic principles, and help you to decide on how to choose a control strategy. It should help you to answer some of the relatively elementary questions which our Customer Support group often get asked, such as:-

Should I use closed loop control?

Should I put in a function generator to linearize the control?

Should I use 3-term control?

CONTROL SYSTEMS ANALYSIS

We have deliberately kept this Section simple, with the minimum of mathematics, and no mention of the classical methods of analysis like Bode and Nyquist diagrams, z-transforms, etc. However, we hope that after reading this Section you will want to carry on learning, get hold of one of the standard textbooks on control theory, and get to grips with the various analytical techniques. You may get by with little knowledge of control systems analysis for most systems, but sooner or later you may come across one which is outside the scope of this book. That's when your knowledge of analytical techniques will stand you in good stead.

----oOo----

Now, in a practical situation you may know already that the control system above is not going to be adequate, and that you will need closed loop control, with a measurement from some sort of flow transducer to check the flow. No matter, always begin by considering open loop control. If it will work well enough, then you've saved yourself the cost of that transducer. You've also avoided having to consider possible stability problems. If it won't work well enough, then you should be able to write down figures for how much improvement your closed loop control has got to give you.

There are two basic factors you should consider when deciding whether open loop control will work well enough or not:-

1. Linearity -
   Assuming that zero setpoint gives zero flow and maximum setpoint gives maximum flow, will you get exactly half maximum flow when the setpoint is half of its maximum value?

2. Disturbances -
   If you leave the setpoint fixed (say half way), are there factors you have not so far considered which will affect the flow, and make the flow you get for a given setpoint vary?

LINEARITY

Linearity is something we can correct for by using the function generator FGEN Special Function S30 in place of LINCON; we can change our rung to:

Fig. 3
Linearized Open Loop Control - GEM 80 rung

We do, however, have a problem in deciding what figures to put in for the table locations following W300. We can only work out these values by a calibration procedure where we try various signal levels (i.e. various numbers) for B7 output, and measure what flow we get for each size of signal. We can then design our FGEN so that, as we vary the setpoint, we get a flow directly proportional to the setpoint.

DISTURBANCES

The second factor we must consider is disturbances. By a "disturbance" we mean anything which may influence what we are trying to control. A disturbance is not necessarily something which is rapid. A very common type of disturbance is ambient temperature variation. Ambient temperature will, in most parts of the world, vary on a 24 hour cycle from a minimum sometime during

the night to a maximum sometime after midday; it will also vary on a yearly winter and summer cycle. Ambient temperature disturbances affect almost all types of control systems. In electrical systems, temperature affects the resistance of cabling and of solenoid and motor windings, and thus alters the current flowing for a given potential difference. In temperature controls, ambient temperature affects heat losses, and thus alters the heat input required to maintain a given temperature. In our example of a flow control, temperature affects the density and viscosity of the fluid, and thus the resistance to flow of pipework and valves for a given pressure or head.

For any open loop system, we need to identify all the possible disturbances we can think of, and then to quantify them. For our example of a flow control, we have already identified one disturbance as being the ambient temperature. Other obvious disturbances are upstream pressure and downstream pressure. If the fluid is not a single liquid or gas but consists of a mixture of liquids or gases, then variation of composition may be another disturbance. If the pipework is not fixed but, for instance, has a number of alternative outlets into different vessels controlled by stop cocks, then the flow for a given valve opening could vary depending on which outlet the fluid was routed to.

All the effects considered above are where the flow can change from disturbances when the valve opening is fixed. However, for a given setpoint, it is possible that the valve opening may not remain precisely fixed; disturbances may cause the valve opening to wander to a minor extent. For instance, temperature changes (yet again) may cause the current from the analog output module to alter slightly. Also, there might be some stiction in the valve actuator, causing the valve to stop short of the desired opening. For the same value of setpoint, we might then get different valve openings depending on whether the particular setpoint value was approached from above or below (i.e. from a higher or from a lower value of setpoint).

In general, for any open loop system, you should be able to identify disturbances of two types:

- Those which affect the output when the controlling device is held in a constant steady-state condition.

- Those which cause changes to the controlling device for a fixed setpoint.

QUANTIFYING THE DISTURBANCES

For our control system, we need to take each of the possible disturbances we have identified and work out (or make some guestimate of) the amount by which the quantity we are trying to control will vary due to that disturbance.

(Of course, you could measure the effects of disturbances by carrying out tests. However, to do this you must have already had the equipment installed. This is usually much too late in the day to start deciding whether you are going to be using open or closed loop control, as you will want to include any measuring transducers as part of the installation, and not add them afterwards. Consequently, where you don't know the effect of a particular disturbance, you will have to make an informed guess.)

Thus, for our flow control example, we might be able to list the effects of disturbances something like the list below:-

DISTURBANCE	FLOW VARIATION
For fixed valve opening:	
Ambient temperature $5^{o}C$-$30^{o}C$	$\pm$2.4%
Upstream pressure $\pm$5%	$\pm$2.5%
Downstream pressure $\pm$1%	$\pm$0.5%
Liquid composition	$\pm$1.4%
Different pipe routes	$\pm$1.0%
Variation of valve opening for fixed setpoint:	
Analog output module	$\pm$0.15%
Valve actuator	$\pm$0.2%
	$\pm$8.15%

For any type of control, you should be able to compile yourself a table something like that above. The question now is: Is the bottom line figure small enough? If, in this example, you only wanted to control the flow to an accuracy of 10%, then the open loop control would be good enough. If, however, you wanted to control the flow to 1% or even better, you would probably have to use a flow measuring transducer and go to closed loop control.

OPEN LOOP COMPENSATION

In theory, you can improve on an open loop system by adding additional signals to the setpoint as in Fig. 4, for instance:

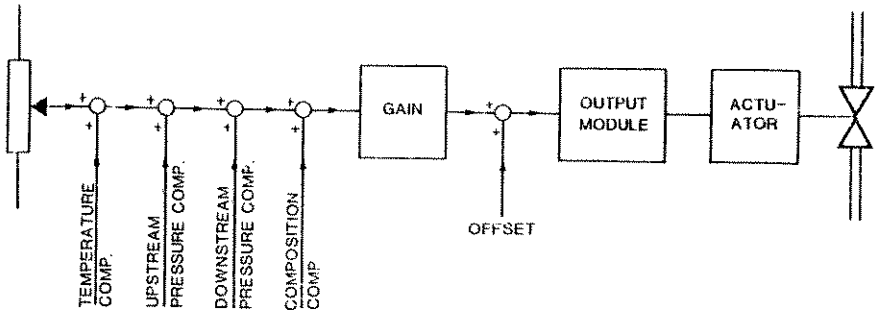

Fig. 4
Compensated Open Loop Control - Block Diagram

In practice, however, such a system is probably uneconomic to construct and impossible to calibrate.

First, we need a signal from somewhere to compensate for each of the disturbances we previously identified and quantified. Where do we get all these signals from? We may need to have several different sensors, whereas, if we put in just one single sensor - a flow transducer - we can compensate for all the disturbances at once, using a closed loop system.

Second, to calibrate the various compensation signals (each of which may need shaping with a function generator), we need to operate the system in such a way that we can introduce each disturbance on its own, without any of the other disturbances occurring at the same time. This may just not be possible. Again, it is much simpler to avoid all these calibration problems by using closed loop control, which requires no calibration (or very little).

SUMMARY ON OPEN CONTROL

1. Always consider open loop control first.
2. Identify all the possible disturbances you can think of.
3. Quantify the effects of these disturbances, and total them up.
4. If the bottom line figure is small enough for the accuracy you require, then use open loop control. If this control will not give a linear characteristic between the setpoint and the output value, then linearize it with a function generator.
5. If, however, the bottom line figure is too large for the accuracy you require, then use closed loop control to compensate for the effects of disturbances.

----o0o----

INTRODUCTION

In closed loop control, the first thing we have got to do is to measure the output we are trying to control. Our system can then determine whether there is any error between our setpoint and the actual value we are getting, and take appropriate action. The block diagram for a closed loop flow control, equivalent to the earlier open loop flow control in Fig. 1, is as below:

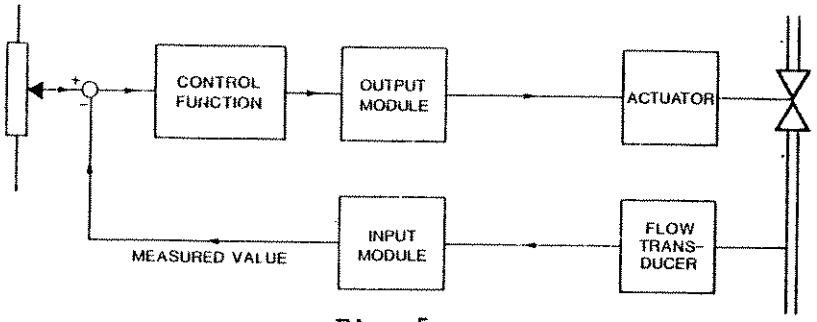

Fig. 5
Closed Loop Control - Block Diagram

It is a most important point (which is a self-evident truth, but people have been known to forget it) that the accuracy of a closed loop system cannot be better than the accuracy of the measurement. Before considering how we design the program for the GEM 80 controller to achieve closed loop control, we must therefore consider each item in the feedback path, up to the point at which our measured value is subtracted, as a number, from the setpoint.

TRANSDUCERS.

If you want a 1% accuracy control, begin by looking for a transducer which is 0.5% accurate or better, i.e. allow for the fact that there are other items in the feedback path with their own inaccuracies.

The transducer you choose must maintain its accuracy in spite of the disturbances which we looked at previously when considering open loop control. Beware of "specmanship" in manufacturers' data sheets and catalogs; you may need to talk to a technical person, not just the travelling sales rep.

In our example of a flow control, we can find numerous types of flow transducers, operating on a wide range of physical principles (e.g. vortex shedding, ultrasonic beam deflection or doppler effect, differential pressure drop across an orifice plate, etc.), some of which will be more suitable for the particular fluid and flow rates than others.

An important factor is the range of flows over which you need to maintain precise control. A transducer may be very accurate at your maximum flow rate, but its accuracy may be quite poor at your minimum flow rate. If you want good accuracy at 100% flow and at 10% flow, then you need a transducer which, in process engineers'

jargon, is good for a "10-to-1 turndown", or has "10-to-1 rangeability".

Whenever you see a percentage accuracy quoted in a catalog or data sheet, you need to ask the question "% of what?" Is it % of maximum flow rate, actual flow rate, or span? For instance, suppose one transducer is quoted as having an accuracy of 0.5% of maximum flow rate, while a second has an accuracy of 1% of actual flow rate, which is the better? The question is answered by tabulating the accuracies at different flow rates as below:

Flow rate (% of max.)	Accuracies			
	First flow transducer		Second flow transducer	
	% of max.	% of actual	% of max.	% of actual
100%	0.5%	0.5%	1.0%	1.0%
50%	0.5%	1.0%	0.5%	1.0%
20%	0.5%	2.5%	0.2%	1.0%
10%	0.5%	5.0%	0.1%	1.0%

For a 10-to-1 turndown, you would probably choose the second transducer, which is clearly better at low flow rates. On the other hand, the first transducer is better at high flow rates, and you would prefer it if you had only a 2-to-1 turndown.

Where percentages are quoted as "% of actual", however, don't expect a transducer to necessarily maintain this accuracy down to zero. A transducer may give a zero offset, i.e. when the flow or other measured quantity is zero, the transducer may give a slight output signal. If it doesn't do this, it possibly gives signals at low levels which are no longer accurate.

Once you are clear as to what percentages mean in a catalog or data sheet, you are now in a position to check how the transducer performs in the presence of those disturbances, taken from the list you made for open loop control (see Page 7), which could influence it. For our example, these would be:

- Temperature. (Some flow transducers don't need any temperature compensation, some do, and others have temperature compensation built-in. With those which need temperature compensation, you might have to fit a separate temperature sensor to provide a separate input signal to the Controller with which you could then compensate for temperature in your GEM 80 program.)

- Fluid pressure

- Fluid composition

These are the basic things to look for as regards accuracy in our particular example. There are, however, one or two practical points, given in the next Item, which you must also consider when choosing a transducer.

SIGNAL CONDITIONING

The output signals from many types of transducer are not suitable for feeding directly into a GEM 80 system.

First, some signals are too small (e.g. from thermocouples). Second, some transducers need special circuitry, such as an excitation supply (for resistance thermometers and strain gauges) or cold junction compensation (for thermocouples). Third, even if the transducer does provide a suitable signal which could be fed directly into a GEM 80 module, you may want to provide isolation between the transducer circuits and the GEM 80 circuits, or between different transducer circuits, to guard against maloperation in case of electrical faults.

For these reasons, you may need to have "signal conditioning" equipment between the transducers and the GEM 80. Signal conditioning equipment is the subject of a GEM 80 Application Data Sheet, and so we do not go into it in detail here.

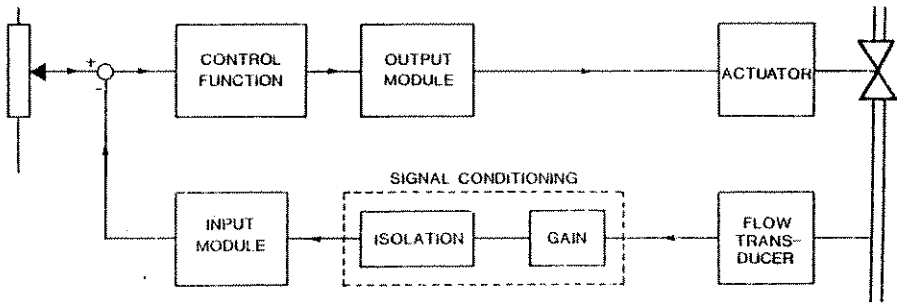

Fig. 6
Signal conditioning between transducer and input module

However, because the GEM 80 does not get its signals directly from transducers where you use signal conditioning, then you have got to choose signal conditioning equipment which, as for the transducers themselves, is sufficiently accurate. Any inaccuracies will add to those of the transducer.

ANALOG TO DIGITAL CONVERSION

Some types of transducer produce digital signals directly, either as numbers, or as trains of pulses (e.g. turbine flowmeter with magnetic pick-off). However, the majority of transducers provide analog signals which need converting into digital signals before they can be handled by any microprocessor system such as a GEM 80 controller. Analog input signals therefore have to be fed to a chip known as an ADC (analog-to-digital convertor), which is included on any GEM 80 analog input module.

ADC's, however, are not absolutely accurate and will give some slight error, which may drift with temperature. The inaccuracies for any GEM 80 analog module are defined in the specifications on the Product Data Sheets for these modules.

Even if an ADC had no inaccuracies at all, it would still introduce a small error due to quantization; that is, the input signal to an ADC has got to change by a certain amount before the ADC will change its output number by 1 bit. The graph of analog input signal to digital output signal for an ADC is a staircase, and therefore the quantization error can be expressed as ±half a bit (see Fig. 7).

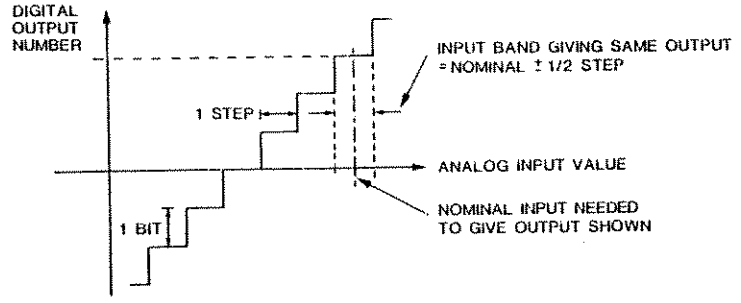

Fig. 7
Analog-to-Digital Conversion

For instance, a 12-bit ADC will give an output number in the range 0 to 4095 (or an equivalent span if it accepts positive and negative inputs), so that the quantization error is ±1/2 in 4095, or 1 part in 8190, or about 0.012% of the span.

LINEARIZATION AND SCALING

The output number from the ADC on a GEM 80 analog input module is not necessarily usable directly as the measured value which can be compared with the setpoint.

Some transducers do not provide outputs which are linear with the quantity being measured, e.g. thermocouples, RTD's, and many types of flowmeter. You have two possible choices here - either to use signal conditioning transmitters which linearize the signals, or else to linearize them within the GEM 80 using the FGEN Special Function. It is more accurate to do it within the GEM 80, because functions provided by the program can't drift with temperature, whereas those provided by analog circuitry generally do. Compare the specs of ordinary signal conditioning transmitters with the specs of linearizing ones.

Even where the ADC output does not need linearizing, it is likely that the numbers the ADC provides are in a somewhat arbitrary range, and do not relate directly to real quantities. It is much easier when monitoring values in GEM 80 tables if the numbers "mean" something, e.g. if values for temperatures can be read directly as ^{o}C or ^{o}F without you having to pull out your pocket calculator to convert them. Therefore you may want to scale the numbers from what the ADC gives into "real" values using the LINCON Special Function. If you are using an FGEN to linearize a signal, then you would similarly arrange for the output from the FGEN to be in real values.

However, it is important that the numbers you get as output from the LINCON, FGEN, or any other conversion function, are sufficiently large not to sacrifice accuracy. If your ADC provides numbers in the range 0 to 4095, say, then a 1 bit change represents 0.025%. However, if you scaled this down with a LINCON function to give you numbers in the range 0 to 200, the smallest change in measured value would be increased to 0.5%. Therefore, rather than scaling values into real quantities, you may have to settle for 10 or 100 times the real quantities. In the example quoted above, you should use your LINCON to convert the numbers so as to be in the range 0 to 20000, rather than 0 to 200. Always make the resolution of the output of the LINCON, FGEN or other conversion function better than that of the ADC.

You will find the resolution of GEM 80 analog input modules given on their Product Data Sheets.

If you are linearizing a signal with an FGEN, then the accuracy will depend on how many straight line segments you use to fit the curve. Don't waste time, however, trying to produce a highly accurate FGEN function if your transducer is not also highly accurate.

QUANTIFYING THE INACCURACIES

We now need to draw up a table of the inaccuracies we will be getting on our transducer signal, in a similar fashion to the way we tabulated the effects of disturbances on the open loop system, something like the list below:

ITEM	INACCURACY
Transducer	0.5%
Signal conditioning	0.3%
Analog input module	0.15%
Quantization	0.012%
Linearization (not required)	0.000%
	0.962%

For any type of measurement signal, you should be able to compile yourself a table something like that above. The question now is: Is the bottom line figure small enough? If the figure is too large, we do not in this case need to alter the control strategy (unlike the case where we looked at disturbances on open loop control). Instead, we have to use higher quality equipment. For instance, if we change from a 0.5% accuracy transducer to a 0.1% one, we will cut down the bottom line figure to 0.562% in the example above.

If you require only a marginal improvement in the bottom line figure, note that the figures you have put in your table are probably taken from manufacturers' data sheets and catalogs, in which case they are probably maximum or worst case figures. You

may be prepared to take a chance on the bottom line figure in a real system being better than this.

However, if you have got to guarantee that the complete equipment will meet a defined performance spec., then you may consider it a good insurance policy to use transducers, signal conditioning, etc., to a better specification, even though the cost is higher.

SUMMARY ON MEASUREMENT SIGNALS

1. Choose a transducer of a type which is sufficiently accurate in spite of the disturbances considered previously when looking at open loop control.

2. Identify other inaccuracies introduced into the measurement by other equipment besides the transducer, up to the point where the signal, in the form of a digital number, will be compared with the setpoint.

3. Make sure you don't introduce further inaccuracies when you linearize or scale the output number from the ADC in the GEM 80 program. Make sure that the number span is capable of a higher resolution than that of the analog input module.

4. Quantify all the inaccuracies, and total them up.

5. If the bottom line figure is too large, go back to your manufacturers' data sheets and catalogs, and choose a better transducer or better signal conditioning equipment, etc.

----oOo----

INTRODUCTION

In this Item, we digress from our main theme of accuracies, and consider a few practical points. We cannot cover all possible process measurements, but give a few of the commoner pitfalls.

SIGNAL CONDITIONING

Many types of signal conditioning transmitter provide single-ended outputs, e.g. positive-only outputs, and not outputs which can swing both positive and negative. With a single-ended output transmitter, its accuracy may deteriorate close to zero output (i.e. instead of the characteristic remaining a straight line all the way down to zero, there is some rounding off as zero is approached). If the type of transmitter you envision using is of this type, then make sure that the span of the signal from it has an elevated zero, e.g. specify it so that the signal into the GEM 80 analog input module will be 2-10V, not 0-10V.

Besides avoiding inaccuracies close to zero, using signals with elevated zero also has the advantage that you can detect for broken connections between the signal conditioning equipment and the GEM 80, or failure of a transmitter. Thus, if the transmitter is calibrated to give a minimum signal of 2V, you can detect anything less than 2V in your GEM 80 program as an indication of a faulty transmitter or a cable fault.

REMOTE ELECTRONICS

You can obtain some types of transducers with built-in electronics, or electronics units which mount locally near to the transducers, such that transducer signals are converted locally to standard levels (e.g. 4-20mA), suitable for direct input to a GEM 80 analog input equipment. Using remote electronics, you may therefore be able to avoid having signal conditioning equipment.

Problems to note with this approach are:-

1. Maintenance.
   Remote electronics may be more difficult to get access to than a centrally stationed subrack of signal conditioning transmitter modules.

2. Hot spares.
   To minimize downtime, you can arrange to have a subrack of "hot spares" of signal conditioning transmitter modules, already powered up and loaded such that they have stabilized in temperature. Then, if a transmitter fails, you can replace it in an instant without having to wait some minutes for this spare unit's temperature to stabilize. You are unlikely to be able to do the equivalent with remote electronics.

3. Signal isolation.
Many types of electronics units which mount locally near the transducer have no power input, and require a low voltage supply (e.g. 24V). You could provide each individual electronics unit with its own power supply, but this would be expensive. If, however, you feed them from a common supply, then an earth fault on one signal line will affect all the signals from all the transducers fed by the one supply. If you ever get a second earth fault, or if any of your transducers are grounded, then you could lose all your transducer signals at once.

On the other hand, you may save on cabling costs with remote electronics, e.g. you won't have to run thermocouple compensating leads long distances, but can use ordinary cable instead.

MEASUREMENT OF ELECTRICAL QUANTITIES

If you are making measurements of electrical quantities, there are one or two things to note.

First, except for pure d.c., electrical signals have waveforms. If you are dealing with a.c., you will generally need to rectify this to provide a d.c. signal which a GEM 80 analog input module can then convert into a number. You can either buy signal conditioning transmitters which do the rectification for you, or you might choose to build up your own circuits.

However, just rectifying an a.c. signal will give you a d.c. signal with a large amount of ripple on it at twice the frequency of the a.c. You will therefore need some sort of filter circuit to smooth it. Again, this filtering might be provided by a signal conditioning transmitter, or it could be part of your own circuitry. You must be careful that any filter circuit you use gives you a signal proportional to the mean, not the peak, of the waveform (unless, unusually, this is what you specifically need in your application).

If you are building your own circuits, you must therefore ensure that any smoothing capacitor has paths of roughly equal resistance for charging and discharging. Use circuits like C rather than A or B in Fig. 8. In A and B, the rectifier gives a low impedance path while the capacitor is charging, but a high impedance path to discharging.

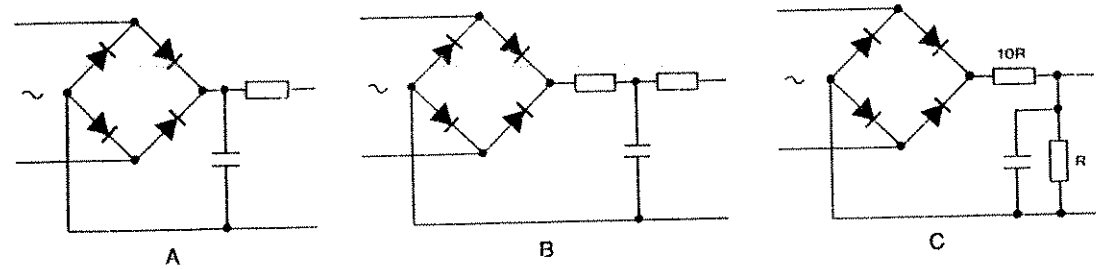

Fig. 8
Filters following rectifiers

Watch out for any similar problems when measuring chopped up a.c. from solid state thyristor or transistor regulators.

If, rather than building up your own circuits, you are using signal conditioning transmitters which accept a.c. signals, then these transmitters will do the filtering, and will generally include precision rectifier circuits. However, calibration may be on the basis of input signals being sinusoidal. If your input signals are not sinusoidal, then check with your supplier on how the transmitters will perform on the particular waveforms you expect to use.

Note that a.c. transmitter input signals will have to be transformed down to a suitable low level, with suitable voltage transformers or current transformers. Depending on the type of transmitter, these may be mounted either within the transmitters or externally to them. If you are using external step-down CT's, make sure that you have some form of voltage limiting devices (e.g. zener diodes) across them, in case you ever manage to open circuit them (e.g. if you unplug a transmitter module).

Note also that, when you rectify an a.c. signal, the diodes used will give a volt drop which varies with temperature. This volt drop variation could affect the accuracy of your d.c. signal. Two ways of overcoming the problem are:

- Rectify at high voltage, where volt drops will be a smaller fraction of the signal. Then potential divide (using a pair of precision low-drift resistors) down to a signal level suitable for the GEM 80 input.

- Use precision rectifiers, using operational amplifiers and diodes (see any op. amp. handbook). This is the way that signal conditioning modules rectify a.c. signals.

In either case, you still need to watch your filtering capacitor charging and discharging paths.

Last, if you require to measure Watts or VAR's in 50 or 60Hz power circuits, it is best to delegate the task of multiplying volts and currents to signal conditioning transmitters designed specifically for the purpose. If you try doing it with a GEM 80 controller, you need very fast sampling, which means having a short program, preventing the full capability of the GEM 80 from being used.

THERMOCOUPLES

If you have not dealt with thermocouples before, you may find them a little confusing.

The way in which a thermocouple works is that it generates an e.m.f. because the tip of the thermocouple is the junction of two dissimilar metal alloys.

However, in a thermocouple circuit, there are other junctions of dissimilar metals, besides at the thermocouple tip itself. All these junctions also produce e.m.f's and, what is more, these e.m.f's vary with temperature, just like that produced by the thermocouple itself. The trick of getting a thermocouple circuit to work properly is to get these other e.m.f's to cancel out in pairs, where possible, and to compensate for any remaining e.m.f. with a temperature sensitive electronic component (usually a simple transistor), so that you are left with just the thermocouple e.m.f. See Fig. 9 below.

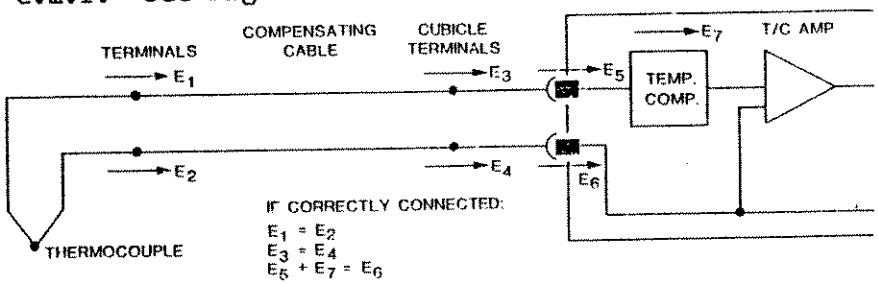

Fig. 9
Thermal e.m.f's in a thermocouple circuit

So, if you want to check the calibration of a thermocouple transmitter, it is no good disconnecting the thermocouple leads and injecting a number of millivolts equivalent to what standard tables tell you a thermocouple would produce at a given temperature. Why? Because either (a) at the points where you are injecting, you have changed the materials from special alloys to the copper used in your test equipment, or (b) if where you are injecting is already copper, you have disconnected subsidiary junctions. You have therefore removed some other e.m.f's besides removing the e.m.f. of the thermocouple itself.

The voltage you should inject must therefore be the voltage the thermocouple would give at the temperature you are trying to simulate, less the voltage the thermocouple would give at whatever temperature the temperature compensation transistor or other component is actually at while you are performing the test. Even so, such a test will not generally be as accurate as using a real thermocouple at a known temperature.

When you install thermocouples (unless you are using remote electronics), the cabling back to your signal conditioning equipment must be run in compensating cable. For the cheaper types of thermocouple (e.g. copper-constantan), the cores of this may be of the same materials as the thermocouple itself, but for the more expensive types (e.g. platinum-platinum/rhodium alloy) the cores will be of different (cheaper) materials. In either case, however, it is essential that you get the cores of the compensating cable the correct way round. If you get them reversed, e.m.f's at subsidiary junctions won't cancel out, and you will get some very odd results. Standard types of compensating cable have standard colour codes for the cores to help you connect them correctly.

TRANSDUCER CHOICE

There are several other factors to bear in mind when choosing transducers besides those given previously. For instance:

- The actual materials of which transducers are constructed can be important to avoid electrolytic corrosion in fluid control systems.

- If you are dealing with fluids such as slurries which are abrasive and can cause wear, you need to check on how the transducer will stand up to the abrasion.

- In some installations, transducers can be subject to vibration. You then need to check whether this will affect their output signals or their life.

- Fitting transducers to your plant may have side effects. For instance, some types of transducer (e.g. orifice plates) give rise to a pressure or head loss across them. Such side effects could influence your choice of transducer type.

Then there are questions on the ease of maintenance:

- What is the anticipated life of the transducer?

- Will its accuracy fall off with time?

- Will it need recalibrating periodically, and if so, how often?

So, although for control purposes you are primarily concerned with accuracy in the presence of disturbances, there are many other practical points to consider when you are choosing the best model of transducer for your particular application. The above list is not comprehensive, and only gives you an indication of the types of factors you should think about.

----o0o----

CLOSED LOOP CONTROL

Having got our measurement signal sorted out, we now proceed to considering the design of the closed loop system. (At least, we have got the accuracy of the measurement signal sorted; we may, however, have to reconsider this signal as regards speed of response as we proceed with the closed loop design.)

SETPOINT SPAN

In Item 2 on "Process Measurements - Accuracy", we mentioned under the heading "Linearization and Scaling" (Page 12) that it was a good idea to scale the number representing the measured value (e.g. using a LINCON Special Function) into "real" units, or a simple multiple of real units, so that it could be monitored directly as a meaningful number. We also said that we should make the span big enough for resolution, so that one bit represented only a small change.

Exactly the same arguments apply to the setpoint. You should therefore make the setpoint span the same range of numbers as the measured value. This ensures that the setpoint resolution is adequate, and at the same time makes comparison between the setpoint and measured value easier.

OPEN LOOP SYSTEM

Before we attempt to close up the loop, we should now have a system as in Fig. 10. If we have scaled the setpoint and measured values to cover the same span of values, then we ought to get, for minimum setpoint, an equal number for the minimum measured value, and, for maximum setpoint, an equal number for the maximum measured value. Because of disturbances, however, we can't expect these values to match exactly, since our open loop control will be subject to the inaccuracies we quantified in Item 1 on "Open Loop Control".

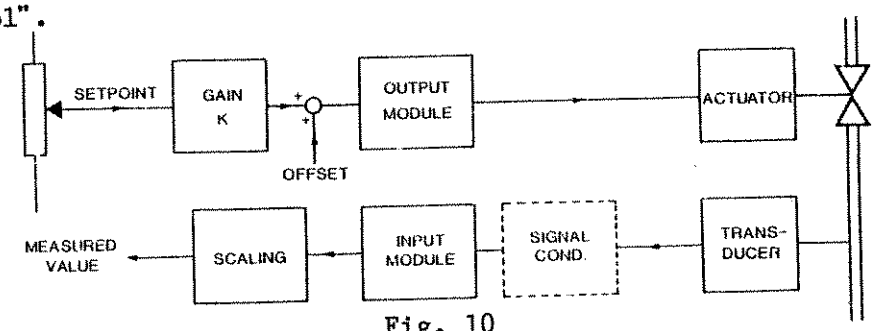

Fig. 10
Open loop control with process measurement added

Suppose that, in our GEM 80 system, the setpoint comes in from input table location A5, and that the measured value comes in from input table location A6. Normally, both of these will need some sort of conversion before they can be used in the control. The setpoint may already be in the right span of numbers, but, for instance, if it comes from thumbwheel switches, then it will need

converting from BCD (Binary Coded Decimal) with the S1 Special Function BCDIN. The measured value, as we saw in Item 2, almost certainly needs scaling and maybe linearizing as well. If we take our original open loop control rung of Fig. 2, what we have now added is two preceding rungs for handling these conversions from unscaled values; for instance:

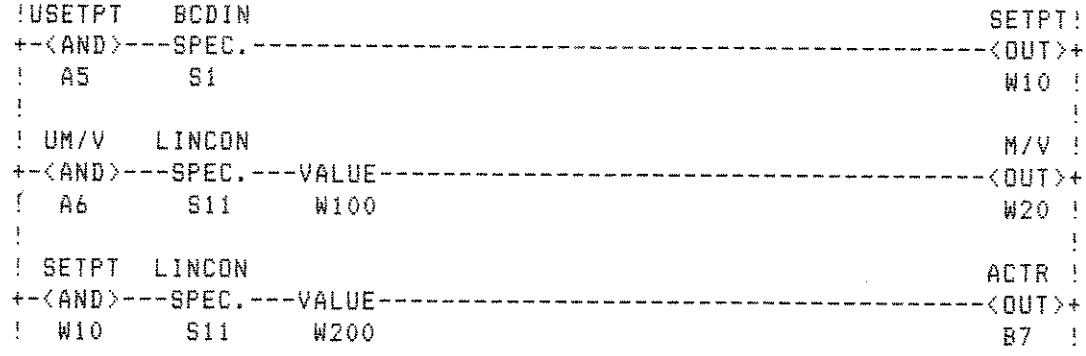

Fig. 11
Open loop control rungs prepared for going closed loop

(The mnemonics used above are SETPT and M/V for the setpoint and measured value after scaling, while USETPT and UM/V mean the unscaled setpoint and measured value.)

We will, in future ladder diagrams, ignore any conversions on the setpoint and measured value, and just use the scaled/converted table values W10 and W20; it will be taken as read that the program contains rungs similar to the first two in Fig. 11 somewhere earlier in the ladder diagram.

PREPARATION FOR GOING CLOSED LOOP

There are two things we must first check out:-

1. The signal polarity of the measured value.
2. What the gain in the open loop control is.

Signal Polarity of the Measured Value

In a closed loop control, we shall be subtracting the measured value from the setpoint to find out if there is any difference between them. Therefore the setpoint and measured values we can monitor, such as W10 and W20 values in the example in Fig. 11, must both be positive. If we find that the measured value is negative, then instead of calculating the difference, we will finish up with effectively adding the two values together, and we get "positive feedback". A system with positive feedback nearly always runs away, the output to the plant increasing rapidly up to the maximum physically obtainable value. You have then lost control of the plant. Try adjusting the setpoint back down to zero and nothing happens. You have to hit the panic button.

It is therefore essential with any closed loop control that you check it out first open loop, as in Figs. 10 and 11, to ensure that polarities are correct. A wrong polarity signal is usually put right simply by swapping over two wires on the transducer, signal conditioning, or analog input module terminals.

Gain in the Open Loop Control

Having got our signal polarities correct, we are now in a position to close up the loop. To do this, we simply need to alter the third rung in Fig. 11 by inserting <SUB> W20. Because the output signal from B7 is directly proportional to the difference between setpoint and measured value, the type of control we have produced is naturally referred to as a proportional control.

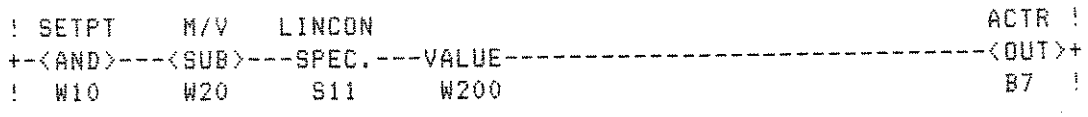

Fig. 12
Closed loop control rung

Now, because we have only a small signal resulting from the subtraction, we need to increase the gain of the LINCON by a large factor.

LOOP GAIN

In the next few pages, we are going to consider:

- how much such a control reduces the effects of disturbances.
- what stability problems we might encounter.

For both cases, the important factor which will keep cropping up in our design formulae is the number of times by which we increase the gain of the LINCON compared with its gain in the open loop control. This factor, which we will denote by M, is called the "loop gain". Thus, if the LINCON in the third rung in Fig. 11 was set to a gain of k, and we increase it for closed loop control in Fig. 12 to k_1, then:

$$\text{Loop gain } M = \frac{k_1}{k}$$

So, before you start experimenting with the LINCON gain in the closed loop control rung of Fig. 12, make a note of what the gain k was for the open loop control rung (the third rung) of Fig. 11, regardless of whether you came by this value from experiment or calculation.

PROPORTIONAL CONTROL

Let's redraw our proportional closed loop control as a block diagram equivalent to Fig. 12:

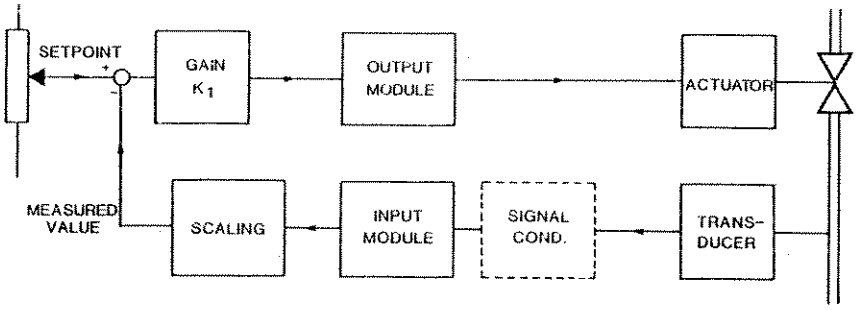

Fig. 13
Proportional Closed Loop Control - Block Diagram

The first important point we should note is that, if we are going to get any output to our valve actuator at all, then there has got to be some difference between the setpoint and the measured value.

As we wind up the setpoint to get a bigger flow, the output to the valve actuator has got to increase to give us this bigger flow, and therefore the difference between setpoint and measured value has also got to increase. This means that, in spite of having taken the trouble to make the setpoint span and the measured value span equal when the control was open loop, we don't in fact get equal values with a proportional control when it is operated as a closed loop as shown in Fig. 13.

For instance, if we had arranged the measured value to span the range from 0 to 10000, we might find that we would have to screw the setpoint up to a small value (let's say 80) before we got any valve movement at all, and that we had to screw it over-range (let's assume 10400) to get the maximum flow of 10000 measured value.

In theory, we could avoid the problem if we could have an infinite gain for k_1. Then, we would only need an infinitesimal difference between the setpoint and the measured value to provide a signal big enough to drive the actuator. In practice, we can't put in an infinite gain, because the system would go unstable. Also, we cannot get an infinitesimal difference between the setpoint and measured value, as the minimum difference we can get in any digital system is 1 bit.

(Later, in Items 9 and 10, we will discover how to get the equivalent of infinite gain using integral control. For the moment, however, we restrict ourselves to finding what we can achieve with a purely proportional control).

Even with a finite gain, however, we can easily get round the problem of the setpoint and measured value not being equal. Instead of subtracting the measured value from the setpoint as in Fig. 13, we can first scale up the setpoint a few percent, add in a small offset, and only then subtract the measured value from it. Our block diagram of Fig. 13 is then modified to become as shown in Fig. 14.

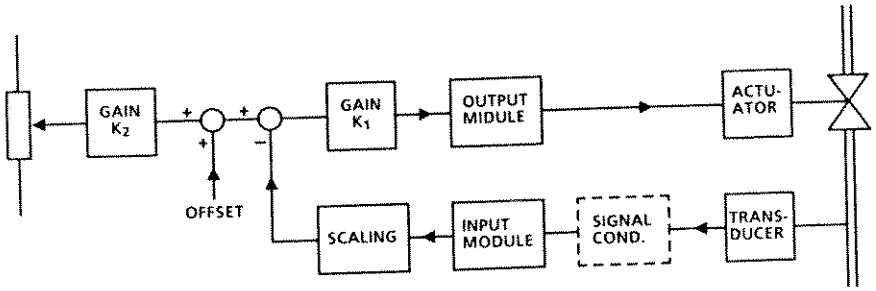

Fig. 14
Proportional Control - Setpoint Scaled Up

For instance, taking the figures of our example, we would need to multiply the setpoint by 1.032. For setpoints ranging from 0 to 10000, the resultant signal would then range between 0 and 10320. We then add in an offset of 80, so that the signal we now subtract the measured value from will range between 80 and 10400. By this means, we have now got back to having both our setpoint and measured values spanning the same number range of 0 to 10000.

In a GEM 80 program, calculations involving gains and/or offsets are simple to do with the LINCON Special Function. We can produce a ladder diagram rung equivalent to Fig. 14 as below:

```
                                                                  ACTR !
! SETPT  LINCON            M/V   LINCON                        <OUT>+
+-<AND>---SPEC.---VALUE---<SUB>---SPEC.---VALUE--------------     B7  !
!  W10     S11     W210     W20     S11     W200
```

Fig. 15
Control rung with Setpoint Scaling

Compared with Fig. 12, all we have done is to insert a LINCON (with its associated VALUE) after the <AND> W10. This first LINCON is used to provide the scaling up of the setpoint and to add in an offset. We then subtract the measured value. After this, we have the LINCON which was already in Fig. 12, whose gain we have increased from its previous value k for open loop control to the new higher gain k_1 for closed loop control.

The required settings of the first LINCON can easily be found by experiment. As in our example, we can initially omit this LINCON from the rung, as in Fig. 12, and subtract the measured value directly from the (not scaled up) setpoint. (We could, instead, insert the LINCON as in Fig. 15 but set it to unity gain). We can then operate the system and find what setpoint values we need for zero and maximum flow. We then work out the gain and offset required, as in our example above, and enter these into the table locations for the first LINCON.

Note that, if we want to monitor the system error in a proportional control system, we have got to put in a separate subtractor,

where we take the difference between the setpoint (not scaled up) and the measured value. In terms of a ladder diagram, we might then have:

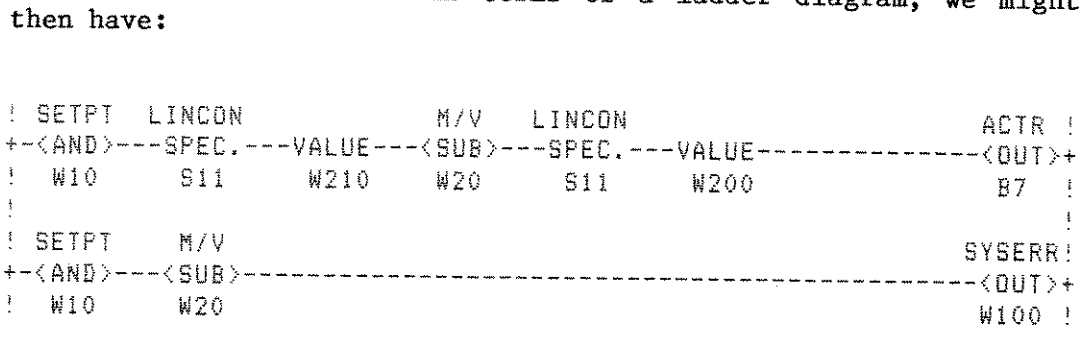

Fig. 16
Proportional Control - Control Rungs

In this section of program, the second rung is just for monitoring the system error in W100. Elsewhere in the program, we don't make use of W100 for control purposes at all.

EFFECT ON ACCURACY OF A CLOSED LOOP PROPORTIONAL CONTROL

The question we have not yet answered, however, is what gain to use for the second LINCON - the one providing the gain k_1. Further, we can't do our trial running to find what values we need for the first LINCON until we have fixed k_1 with the second LINCON.

For the second LINCON, what is important is not the value of k_1 in absolute terms, but, as we said on Page 22, what the loop gain M is. That is, what is important is how many times bigger k_1 is than the gain k required for open loop control, for the same span of setpoint values.

The effects of having a loop gain of M for our closed loop control are:

- **The effects of disturbances are reduced by the factor M+1 compared with their effect on an open loop system.**
- **If our open loop system has a non-linear characteristic between setpoint and output, the linearity will be improved by M+1 times for the closed loop system.**

These two clauses both imply that the higher we can make the loop gain M the better. For a given accuracy, the first clause tells us what the minimum loop gain must be. If we can make M higher than this, we've got a better accuracy than our accuracy spec demands.

CONTROLS NEEDING LINEARIZING

The second clause means that we may be able to dispense with having to put in any FGEN where one was needed with an open loop system. If the characteristic of the closed loop system is still

not as linear as we would like, then we can still use an FGEN, but it will be a very much simpler one with fewer segments, making calibration correspondingly much easier. The placement of this FGEN, and possible stability problems, are dealt with later in Section 2, in the Item on "Parameter Variation".

WORKED EXAMPLE

Let us now do some sums on what loop gain we would like to have, using our previous example of a flow control.

In our original open loop example, we totalled up all the effects of disturbances we could think of, and arrived at a bottom line figure of 8.15% (see Page 7). We then went on to work out the accuracy of the flow measurement signal, which came to 0.962% (see Page 14).

Let's assume we are after a 0.5% overall accuracy for the system. The first thing we have got to do is to make some improvement in the measurement signal. Suppose we change our choices of transducer and signal conditioning equipment to 0.1% accuracy types. This will knock our bottom line figure for the flow measurment down to 0.362%, leaving us 0.138% to play with. To reduce the effects of disturbances from 8.15% down to 0.138%, we therefore need a loop gain given by:

$$\frac{8.15}{M+1} = 0.138$$

or:
$$M+1 = \frac{8.15}{0.138}$$

$$= 59.06$$

or:
$$M = 58.06$$

Hence, to achieve an overall accuracy of 0.5% for the system, we need a minimum loop gain of the order of 60.

STABILITY

In this Item, we have stated what the effects of closed loop proportional control are, and done some example calculations. We worked out what loop gain M we would need to meet a specified control accuracy for given sizes of disturbances.

What we have not yet established, however, is whether a control with this loop gain would work satisfactorily. It might be unstable. The next Item will help us resolve this question.

SUMMARY ON THE ACCURACY OF PROPORTIONAL CLOSED LOOP CONTROLS

1. The total accuracy of the system is the sum of two parts:
   a. Accuracy of measurement
   b. Effects of disturbances

2. The accuracy of measurement must therefore be somewhat better than the total accuracy you are after. If it isn't, you will have to choose better equipment for the measurement.

3. Subtract the accuracy of the measurement from the total system accuracy you require. The resulting figure is how much you can allow for the effects of disturbances.

4. Knowing what the effects of disturbances would be if the system were open loop, and what the allowable effects of disturbances are, divide the first by the second. The result is the M+1 you need, i.e. the loop gain + 1.

----oOo----

INTRODUCTION

From our sums on accuracy, we know what loop gain we would like our proportional control to have. However, there is one fly in the ointment - stability. If we just set the gain of our LINCON to the desired value, we may find if we are lucky that the system behaves well, but if we are unlucky it will go into oscillations. How can we tell what is going to happen? And, if the system is unstable, what can we do to make it stable and yet to give us the required accuracy?

The first course open to us is to carry out a mathematical analysis of the system. This is the way in which most textbooks on closed loop control theory expect you to proceed. Unfortunately, you can only carry out such analysis if all the data you need about the system is available. This is possible for some electrical systems, but for mechanical and fluid control systems it is usually not.

OPEN LOOP RESPONSE

So let us adopt an experimental approach. Taking our lead from the way we considered system accuracies, let us begin by studying the response of the very first type of system we considered - the open loop system. Let us run the system without any feedback from the measuring transducer, with a ladder diagram control rung like Fig. 2 or Fig. 3. To measure the response, we can connect a pen recorder with two channels; on one channel we record the measured value signal, and on the other we record the setpoint. We now run the system open loop with the setpoint at, say, 50%, and wait for everything to settle down. When everything seems steady, we start the recorder, and then increase the setpoint by a small amount (e.g. by 5% or 2% of full span) as a step change, and take a recording of the response of the measured value coming back from our measuring transducer via any signal conditioning, etc.

For 95% or more of all systems you will ever come across, the response will look something like that below:

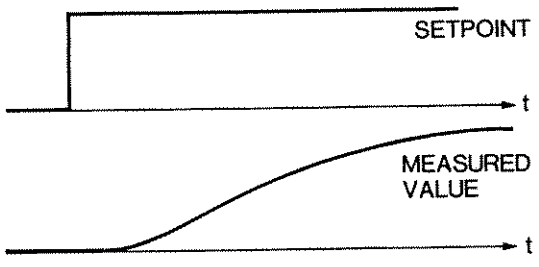

Fig. 17
Open Loop Plant Response

The shape of this curve can be approximated to a time delay and a time constant. The way in which we measure these values off our recording is as follows:

1. Draw straight lines through the initial value and the final value of the transducer signal parallel to the time axis (i.e. parallel to the edge of the recording paper).

2. Draw a straight line as a tangent to the response curve, and see where it cuts the parallel lines at B and C.

3. Mark off on the initial value line the time at which the setpoint changed at A.

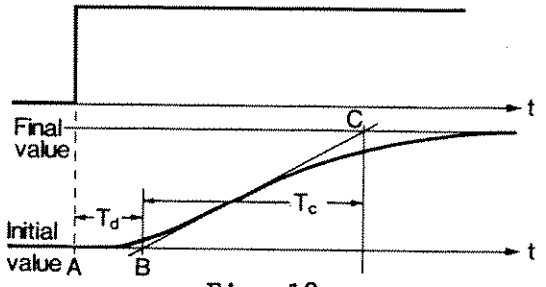

Fig. 18
Measuring Plant Time Delay and Time Constant

Then:

The effective plant time delay T_d is the time from A to B.
The effective plant time constant T_c is the time from B to C.

CLOSED LOOP RESPONSE

What control theory tells us is that, if we operate the system as a closed loop with loop gain M, then:

- **The effective system time delay is the same as the effective plant time delay of the open loop system.**

- **The effective system time constant equals the effective plant time constant of the open loop system reduced by the factor M+1.**

- **The highest loop gain which can be used while maintaining a reasonably good, stable response, is that which makes the system time constant roughly equal to the system time delay.**

The second and third clauses tell us that the maximum loop gain we can use while maintaining a good response is of the order of:

$$M+1_{max} = \frac{T_c}{T_d}$$

where T_c = effective system time constant
T_d = effective system time delay

or:

$$M_{max} = \frac{T_c}{T_d} - 1$$

We will consider later what we need to do if M_{max} is less than what we need to get the accuracy we require. However, let's first look at the sort of responses we might get if we set the system up

to run in closed loop mode, using the control rungs of Fig. 16 for our ladder diagram, with various gains for the second LINCON.

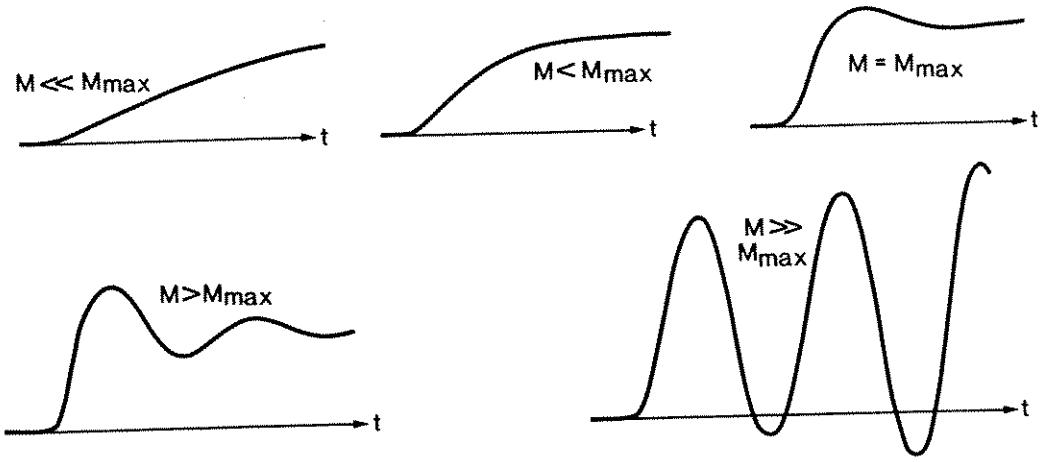

Fig. 19
Variation of Response with Loop Gain M

These diagrams are included just to show you the sort of change in response you can expect to get by altering the loop gain, and how the system can go unstable if M is set too high.

LOOP GAIN SUFFICIENT FOR ACCURACY

If the value you need for accuracy is less than M_{max}, then in most cases you might as well wind the loop gain up nearer to M_{max}, as this will give you both a faster response and a better accuracy.

LOOP GAIN TOO SMALL FOR ACCURACY

Now let us look at the problem we have been putting on one side - what to do if the loop gain we can get with a reasonable looking response is not big enough to achieve the accuracy we want. We will look at three alternative methods of solution.

1. Review our accuracy requirements. We noted that the total system accuracy was made up of (a) accuracy of measurement, and (b) effects of disturbances. What closed loop control does for us is to reduce the effects of disturbances, compared with the open loop system, by M+1 times. Therefore, if we can improve the accuracy of measurement, we can get away with allowing the effects of disturbances to be bigger. We would not then need to make M so big.

Practically, we may not be able to achieve much this way. We may have already chosen equipment for the process measurement which is as accurate as we can afford, or as accurate as is available in the marketplace. Even if we had equipment which was 100% accurate, we might still not be able to use a high enough loop gain M to reduce the effects of disturbances as much as we want. Still, it is worth doing a quick check on whether anything can be achieved this way.

2. Look at the plant we are trying to control, and see whether there is anything we can do about the effective system time constant T_c and time delay T_d, because it is the ratio of these two values which determines the maximum loop gain M we can use before the response starts to become oscillatory. If we could make T_c larger and T_d smaller, we could use a higher loop gain M.

3. Use an integral term, turning the control into a 2-term proportional + integral one (or a PI control for short). This type of control is the subject of later Items. We should note that it consists of an integral control added on to a proportional control. We shall see that the integral term looks after some of our accuracy problems, but it does absolutely nothing to improve the speed of response; in fact it makes it slightly worse.

We still therefore need to look at Method 2, to see if we can increase the loop gain M, if only to improve response. We therefore need to answer the question: What do T_c and T_d physically represent? Where do they arise from in the plant we are trying to control?

EFFECTIVE PLANT TIME DELAY

What we measured off the response recording of the open loop system (Fig. 18) as the effective system time delay is made up of two items:

1. Pure time delays - sampling delays, transport lags, etc.

2. Several small time constants - electronic filters in transducers, actuators, signal conditioning modules, GEM 80 analog modules, etc.

If you were doing a strict mathematical analysis of a system, then you would probably not include item 2 above, but represent all these small time constants separately. However, for our purposes, it is sufficient to treat them all as a single time delay equal to the sum of the individual time constants.

Transport Lags

Of the time delays arising from any plant, the biggest enemy to a fast response is the transport lag. This is where, when we make a change to the control signal to the plant, the output may start to change immediately, but our measuring transducer doesn't see anything till sometime later. For instance, suppose we have a temperature control, and suppose we change the heat input to the boiler or heat exchanger. If we have placed our temperature transducer a long way downstream, then there will be a delay before our transducer notices any change in temperature.

To reduce the transport lag, we should therefore place the trans-

ducer in the outlet pipe as close to the boiler or heat exchanger as possible. If we place it a long way downstream, then our effective time delay T_d will be long, therefore M_{max} will be restricted, and we won't be able to get a fast response or a high accuracy proportional control.

In a heating system, you might argue that you're not really interested in the temperature just after the boiler or heat exchanger, but in the temperature at the outlet from the pipe, and therefore you need your temperature transducer way downstream. In that case, you may need two transducers, one directly downstream of the boiler or heat exchanger and one at the outlet. We will consider this type of problem later in Section 2, under the Item on "Cascade Controls".

Another common type of transport lag is that found with conveyor belt systems, where at one point you control the amount of material being deposited on the belt, but you only measure it (e.g. with a belt weighing transducer) some distance downstream. Again, if you want a fast and accurate control, you should put the belt weigher as close as possible downstream of where you place material on the belt, so as to minimize delays.

The size of transport lags therefore depends on the design of the plant. You may be able to minimize them by the placement of transducers. You may also, knowing the design flow rates, be able to calculate what they are likely to be. This will then give you an idea of what the fastest response is that you should be able to get in practice out of your closed loop control.

If you want a faster response than this, then bad luck - you will have to modify the design of the plant. Don't expect a controller (of any sort, whether analog or digital, whether dedicated to a single control loop or handling several control loops, whether GEM 80 or of some other manufacture) to compensate for the shortcomings of a poorly designed plant. Also, any improvements you can get by using clever control algorithms in place of simple proportional, PI or PID controls will be only marginal.

Sampling Delays

A programmable controller such as a GEM 80 does not look at the measured value continuously, but only scans it at regular intervals. Similarly, it does not update the value of the control signal to the actuator or other controlling device continuously but at regular intervals. (At least, the intervals ought to be regular; it is possible for them not to be regular if you have programmed your GEM 80 incorrectly, in which case your control system might exhibit somewhat random behaviour. We will assume, however, that you have arranged for the scan to be regular by correct programming.)

The effective time delay introduced by sampling the measured value is half the sampling interval. This can be seen from Fig. 20.

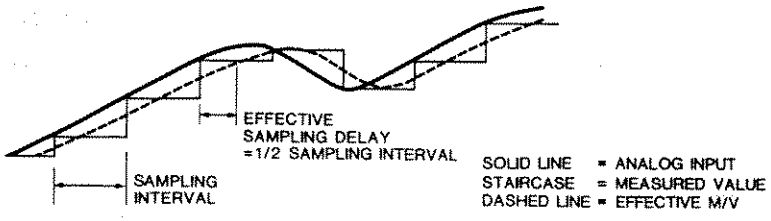

Fig. 20
Delay introduced by Sampling

In addition to this sampling delay, there is also a delay of one program execution, between the GEM 80 controller reading the measured value as an input, and setting the signal to the actuator as an output.

To set the program execution time, you have to preset a value in P-table, in location P1. Where the control is to have as small a delay as possible, you will want to execute your control rungs on every program cycle, and not put them in a BLOCK. In this case, the effective delay produced by the GEM 80 will be 1.5 times the program execution time.

However, if you do this, you will find that the controller delay comes out very much shorter than the plant dead time, for the majority of systems. (There are a few types of system where this is not so; see Item 8 on "Special Cases" later.) In fact, for most systems, the question is not how short you can make the sampling delay but how long you can make it. If your program services every control loop on every program cycle, it can be doing unnecessary calculations and make the program cycle very long. If, therefore, you have several control loops, it is normally better to arrange your program to service not more than one control loop on each program cycle, and to service the next control loop on the next or on a later program cycle. To do this, you need to place your control rungs for each loop in their own BLOCKs, each of which is obeyed at a regular time interval.

In this way, the program loading on successive cycles is evened out, resulting in a faster program cycle time, but making the sampling interval for any control loop longer. The total delay introduced by the GEM 80 controller then equals:

- half the BLOCK execution interval, plus:

- one program execution time.

For instance, if you operate the BLOCK containing the control rungs for a particular loop at 1 second intervals, and if the program cycle time is 20 ms, the effective time delay will be 0.5 + 0.02, or 0.52 seconds.

For most systems, you are not going to notice much practical difference in the system response if the effective system time delay is made 10% or 12% longer than the effective plant time delay. Therefore the sampling interval can often be made up to 20% or 25% of the effective plant time delay (because of the halving effect shown in Fig. 20).

Small Time Constants

The various small filtering time constants which we treat together as an additional delay are there for a purpose - to prevent spurious effects from electrical noise.

Remember that a microprocessor-based controller such as a GEM 80 samples signals. It takes "snapshots" of each input signal at regular intervals. If your signals are not clean but have spurious ripple on them, then one snapshot could be taken when the ripple was at a peak, while the next snapshot could be taken when the ripple was at a trough (see Fig. 21).

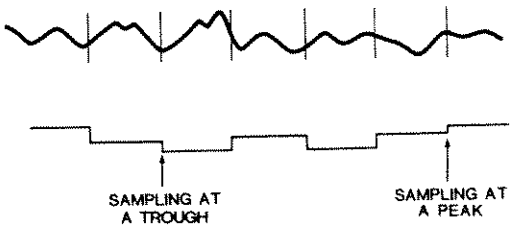

Fig. 21
Effect of sampling a noisy signal

Consequently, if you monitored the value of a signal in a GEM 80 input table, you might find that it was fluctuating in a somewhat random manner. If you want to reduce such fluctuation, then you have either got to use extra filtering or else get rid of the ripple. The first course of action would increase the size of the effective system time delay, and could make the closed loop response slower than you would like. The second course of action is to see whether you can identify the source of ripple, and reduce the magnitude of the ripple by means other than filtering. We will look at this in the next Item.

EFFECTIVE PLANT TIME CONSTANT

In the majority of cases:

o **A time constant represents storage with some losses.**

For instance, if you apply heat to a vessel, the temperature of the contents will increase. However, as the temperature increases, the heat losses also increase so that, instead of the temperature continuing to rise indefinitely, the temperature gradually levels off. The vessel eventually reaches a steady state temperature where all the heat input goes into heat losses.

We get a curve, if we plot temperature against time, as shown in Fig. 22.

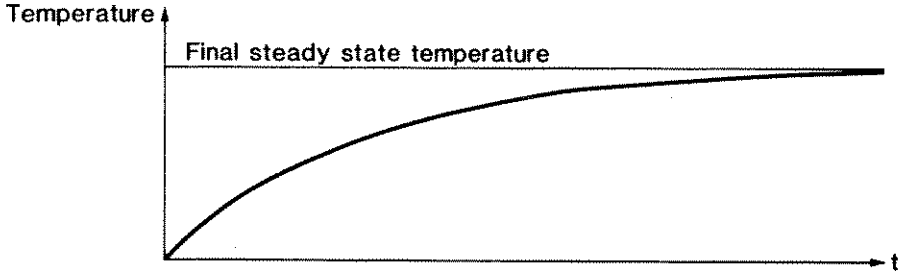

Fig. 22
Temperature of a Heated Vessel against Time

(This assumes that we are not putting so much heat into the vessel that its temperature or pressure get too high for safety.)

Similarly, if you apply a d.c. voltage to an electromagnet (e.g. a solenoid coil or a motor winding), the current will start to increase as energy is pumped into the iron of the magnetic circuit.

However, as the current increases, you get a voltage drop because of the resistance of the winding. Instead of the current continuing to rise indefinitely, the current gradually levels off, and reaches a steady value where all the input power (VI) goes into providing the losses (I^2R). If we plot a curve of current against time, we get exactly the same shape of curve as for the curve of temperature of a heated vessel:

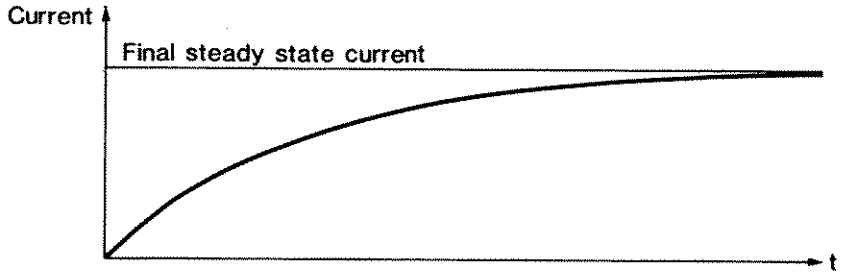

Fig. 23
Electric Current in a Coil

(Note: The above curve assumes that the current is not sufficient to make the iron in the magnetic circuit saturate, and that you can therefore treat the inductance in the electrical circuit as a constant.)

In both of the above examples, reducing the losses will make the time constant longer. You can then use a bigger loop gain M, giving a more accurate proportional control. However, the speed of response will remain roughly the same. If you double the time constant, you can get roughly twice the loop gain M. This would have made the response twice as fast if you hadn't altered the time constant. However, since you doubled it, the two effects

cancel out, leaving you with a response which is virtually unchanged.

In practice, reducing the losses in both these examples would cost money. For the heated vessel, you would have to thermally insulate it better. For the electrical coil, you would have to use a winding with more copper to give a lower I^2R loss. The length of time constant is therefore a measure of how energy efficient the plant is. The longer the time constant, the more energy efficient the plant. In such cases, the time constant of the plant is something which we cannot easily modify. It is determined by economics, by how much we are prepared to pay in capital cost to reduce the running costs in energy losses.

In real plant, time constants also arise where the "losses" are of our own making. For instance, if we have a heat exchanger vessel, we put heat in, as in Fig. 22, but the fluid we are heating, instead of remaining fixed in the vessel, is in continuous flow. The vessel eventually reaches a steady state temperature where, as before, all the heat input goes into heat "losses". The heat loss, in this case, primarily consists of the heat content of the fluid output from the heat exchanger. If you alter the flow rate of the output fluid, you alter the time constant. The time constant will be longest when the flow is low, and will become shorter if you increase the flow. However, changing the flow also changes the gain. That is, if you think of the heat input being controlled open loop, you have to put more heat in for the same temperature rise of the output fluid. This type of system, where time constants, time delays and gains vary with operating conditions, is dealt with in more detail in Section 2, in the Item on "Parameter Variation".

Now, let us be clear about the difference between a time constant and a time delay. The curves in Fig. 22 and Fig. 23 show the way in which the output from a time constant responds to a step change of input. When the input is changed, the output starts changing immediately, but it takes a long time to reach its steady state value. With a time delay, nothing at all happens when the input is changed until after the delay, when the output reaches its final value instantaneously (see Fig. 24):

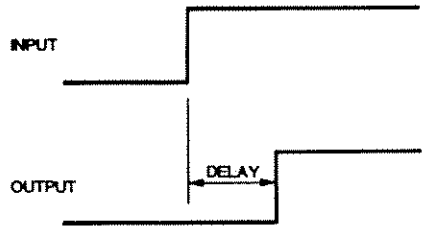

Fig. 24
Response of a Time Delay

Slow Transducer or Signal Conditioning

It is possible to get an open loop plant response like that shown in Fig. 17, but where the effective system time constant comes, not from the plant, but from the transducer or other equipment in the process measurement path.

Now, although we might be able to get a respectable looking response for the measured value out of such a system, the actual process variable we are trying to control might be doing something entirely different. If our transducer or other equipment in the measurement path is so slow that it provides the major time constant in the system, then we haven't really got a proper measurement at all.

The only time constants we can accept in the measurement path are small ones (like filters) which can be treated as part of the effective plant time delay. If we have a transducer or other equipment in the measurement path providing a significant time constant, then the plant variable will tend to overshoot on any change of setpoint, even though the measured value seen by the GEM 80 controller is perfectly well behaved.

So, we have yet another thing to check on our process measurements. Besides the accuracy (Pages 9ff) and a number of practical points (Pages 15ff), we need to check that the transducers and signal conditioning equipment have a fast enough response themselves, which must be much quicker than the plant we are trying to control.

SUMMARY ON THE RESPONSE OF PROPORTIONAL CLOSED LOOP CONTROLS

1. For the majority of systems, the open loop response is effectively a time delay and a time constant.

2. Closed loop control makes little difference to the effective system time delay compared with open loop control, but reduces the effective system time constant M+1 times, where M is the loop gain.

3. To maintain a reasonable response, M should not be made larger than the value which would make the effective system time constant equal to the effective system time delay.

4. It is theoretically possible to achieve a higher loop gain M if either the system time delay can be made shorter and/or the open loop time constant can be made longer.

5. In some cases, you may be able to minimize the time delay by placing measuring transducers a shorter distance downstream of the process, thus avoiding long transport lags.

6. In most cases it is not possible to lengthen the effective system time constant, neither is it possible to reduce the size of small filter time constants which are treated as part of the effective system time delay.

7. If the loop gain achievable with a good response is insufficient for the accuracy requirements, then proportional+integral (PI) control will be required.

-----o0o-----

INTRODUCTION

This Item represents another digression from our main themes of accuracy and response, which we will pick up again in the next Item.

In the last Item, we mentioned under the heading "Small Time Constants" that filters were included in electronic equipment to reduce the effects of electrical noise. Such noise could produce fluctuations in our process measurement signals. We said that, if we wanted to reduce the filter time constants so as to reduce the effective system time delay, then we would need to look at ways of reducing electrical noise, other than filtering.

EXTERNAL ELECTRICAL NOISE

Some types of electrical noise come from sources which are entirely isolated from our control and measurement cirucits. This type of noise is often referred to as "interference". The noise gets in by stray capacitance (electrostatic coupling), transformer action (electromagnetic coupling), or as radio waves (where our circuits act as aerials).

The first thing to check is cabling. All the above forms of interference can be reduced by physically increasing the distance between the noise-source circuits and our control and measurement circuits. The GEM 80 Installation & Maintenance Manual (Publication T291) gives some recommended segregation distances between different types of cabling.

Second, capacitive coupling can be reduced by using screened cable, and magnetic coupling can be reduced by twisted-pairs for the go and return wires carrying any signal. Note that normal practice is to use such types of cable on signal wiring only. However, because coupling effects are mutual, you can also reduce capacitive coupling by screening the source of noise, and reduce magnetic coupling by twisting the go and return wires of power circuits.

Third, some types of noise can be suppressed at source. You obviously cannot eliminate 50 or 60 Hz ripple on power wiring, but you may be able to suppress the noise which, otherwise, would be generated transiently when contacts open to break inductive circuits (e.g. contactor or solenoid coil circuits), by fitting suitable RC (resistor-capacitor) snubber circuits across the coils.

However, interference can also be caused by resistive coupling with non-isolated circuits. If you cable your circuits so that several of them use a common return wire, you may get the volt drop in this wire changing when the current through another circuit alters. Wherever possible, avoid this by providing an individual return wire for each signal.

Last, problems can be caused by earth loops where a particular circuit is earthed at more than one point. Note that two physical earths (earthing rods or plates buried in the ground) are never at the same potential, and multiple earths can cause spurious currents to circulate.

The above paragraphs are a very brief summary of good cabling practice to avoid spurious electrical noise. We haven't got space in a manual on systems design to go into the subject in greater detail. Keep a look out for articles in the technical press - cabling practice is a recurrent subject.

INTERNAL ELECTRICAL NOISE

Not all electrical noise, however, comes from outside our system; noise can come from the equipment itself.

First, the output we are trying to control may have ripple on it even before we measure it. This is most common in electrical systems. We made some mention of filtering electrical signals earlier on Page 16.

Second, the measuring transducer may introduce ripple. For instance, any d.c. tachogenerator measuring the speed of a shaft will produce not pure d.c. but a signal with superimposed speed-dependent ripple.

Third, signal conditioning transmitters whose inputs are d.c. generally provide isolation by:

- chopping the input signal into some form of a.c.
- feeding this a.c. via a transformer or opto coupler
- rectifying the a.c. back to d.c. on the GEM 80 side of the coupler.

The output from such transmitters usually has ripple at a frequency dependent on the chopper. To reduce this ripple to an acceptable level, transmitters generally include filtering on their output circuits. The time constants of these filters are usually a compromise. If you want to reduce the magnitude of ripple to a lower level, and so reduce random fluctuation in the signal sampling (see Fig. 21), then you need long time constant filters; if you want a fast response and short time constant filtering, then you have got to accept greater sampling fluctuation.

The only way to reduce the size of filtering without increasing the ripple content is to use better equipment, such as:

- signal conditioning modules which operate at a higher chopper frequency
- tachogenerators with lower ripple content in their outputs
- etc.

Watch out, however, when choosing signal conditioning transmitters working at a higher frequency, that they are sufficiently accurate. You may find that, while you have gained on less ripple and/or a faster response, you have lost out on accuracy or temperature stability.

FILTER COMPONENTS

Although we have described certain things you can do to reduce electrical noise which might allow filtering to be reduced, in practice most filtering is already built into the equipment you buy. You will not generally want to tamper with this, as it will affect its warranty. However, if you have not given any consideration to noise in the first place, you might find yourself having to tack on extra filter circuits externally, mounted on terminal blocks or in odd corners of the installation, to get noise down to an acceptable level. If you have thought about noise, then you will be in a better position to:

- choose the best equipment for your application, given the alternative models available which provide different amounts of filtering and ripple content. For instance, with GEM 80 fast I/O analog input modules, you have the choice of ordering them with 0.5 ms or 5 ms input filters.
- get it cabled up well.

SUMMARY ON ELECTRICAL NOISE

1. Avoid external electrical interference by:
   (a) good cabling practice
   (b) suppression of noise sources

2. To reduce internal electrical noise, choose transducers with a lower ripple content on their output signals, and signal conditioning equipment operating at higher chopper frequencies.

----oOo----

INTRODUCTION

In our discussion of proportional closed loop control stability, we have only so far considered what happens when we make small step changes in the value of the setpoint. We now know, for instance, how we can get a system which oscillates if we make the loop gain M too high. But we have also spent some time discussing how to make M as high as possible to get a faster response. Why do we need a faster response? In many practical cases, we never make rapid changes to the setpoint, so why bother?

The answer is that the response to a step change of setpoint also gives us some idea of how the system will respond if there is a step change disturbance. Remember that the reason for using closed loop control in the first place was to reduce the effects of disturbances. What we calculated in Item 4 on "Proportional Control Accuracy" was, basically, the loop gain M needed to give us the accuracy we required, in spite of disturbances, in the steady state. Obviously, we must also consider what happens transiently. If there is a sudden disturbance to the system, then the system error can suddenly become bigger, going outside the accuracy spec. we require. We therefore need our closed loop control to reduce the system error as quickly as possible and bring it back within spec.

RESPONSE TO AN INPUT ANYWHERE IN A CLOSED LOOP

For any closed loop system, regardless of whether it is a proportional or some other form of control:

- **If a signal is introduced which adds to or subtracts from one of the signals flowing around the loop, the response of the signal at the point of introduction is the same, no matter where the point is.**

Consider the block diagram below:

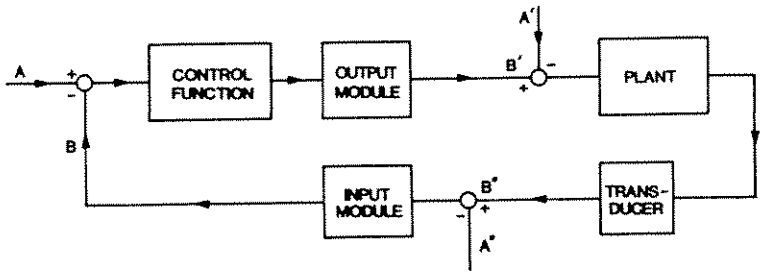

Fig. 25
Disturbance inputs to closed loop controls

We have so far been considering just what happens to signal B (the measured value) when there is a step change in signal A (the scaled setpoint). What the clause above says is that signal B' (the output from the GEM 80 analog module) responds in exactly the same way if we introduce a disturbance A', or the signal B" (the output from the measuring transducer) if we introduce a step disturbance A".

For instance, suppose that we have a temperature control on a vessel, where our closed loop control is adjusting the heat input to the vessel so as to maintain the temperature in accordance with our setpoint setting. We will assume that the predominant time constant in the system is the vessel itself, and that its response to a step change of heat input is therefore as we drew in Fig. 22. Now suppose that there is a step change in heat losses (this might be caused, for instance, by the sun going behind a cloud). All of a sudden our heat input and our heat losses are no longer in balance. Our closed loop system will respond, however, in exactly the same way as the temperature would to a change in setpoint, except that, in this case, it is the heat input responding to a change of heat losses. The speed of response, and the response shape, are the same in both cases:

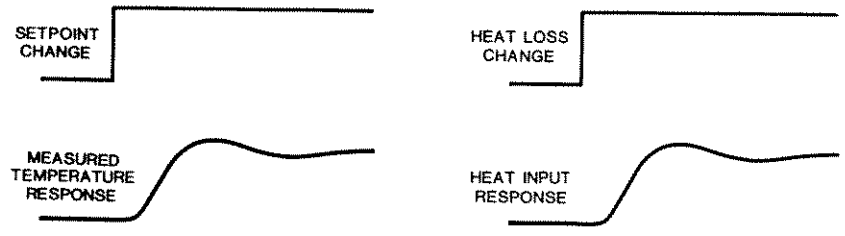

Fig. 26
Response to a step change disturbance

If we subtract the heat input from the heat losses, we get the shortfall in heat input. This will cause the temperature to drop for a time, and then recover. If we consider two systems, one of which is twice as fast as the other, then the transient temperature drop will be approximately half for the faster system:

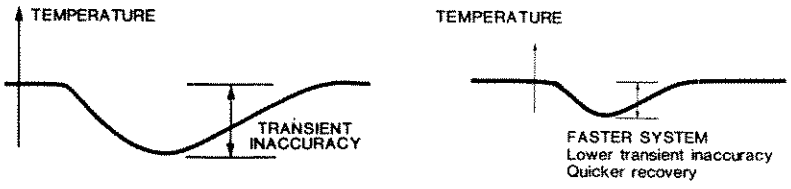

Fig. 27
Transient inaccuracy with a step change disturbance

This, then, is the explanation of why we want our closed loop controls to have a fast response - to reduce the transient effects of disturbances - even though we may never make a rapid change in the value of the setpoint.

----oOo----

INTRODUCTION

In Item 5 on "Proportional Control Stability", we said that, in 95% of the cases you would come across, the open loop response of the plant would be as in Fig. 17, which could be approximated as shown in Fig. 18 to a time delay and a time constant. From time to time, however, you may come across a system where the response is a limiting form of these curves, and the normal rules we have been using for a proportional control can't be applied.

CASE 1: NEGLIGIBLE TIME DELAY

If you try recording a response curve of the open loop system, and you find that the ratio of the time constant to the time delay is very large, you may not be able to measure any noticeable delay on your recording:

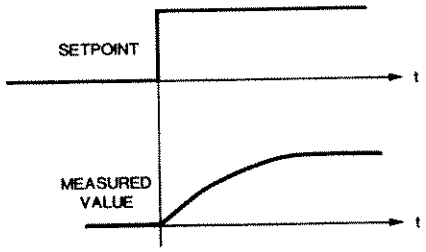

Fig. 28
Response curve with negligible time delay

If in fact there were no time delay at all, then you would be able to make the loop gain M infinite. However, don't forget that there is always a sampling delay with a digital controller like a GEM 80, dependent on how frequently your program services the output to the plant and looks at the measurement signal coming back from the plant, plus a delay of one program execution between the GEM 80 reading the inputs and setting the outputs. Hence, there is an upper bound on what M can be, equal to the ratio of the plant time constant to the sampling + execution delays. However, you should start by making M somewhat less than this because, although not measurable on your recording, there may be other minor delays in the system.

Now, since the effect of closed loop control is to reduce the time constant M+1 times but have little effect on the time delay, you can then take another recording, this time of the closed loop system response, at a higher paper speed. Because the time constant and time delay will now be closer together in size, you will be able to measure the time delay off this recording more accurately (you needn't measure the time constant). Knowing the open loop time constant from your first recording, you can now calculate the maximum value for M somewhat more accurately, and set up the proportional gain accordingly.

CASE 2: NEGLIGIBLE TIME CONSTANT

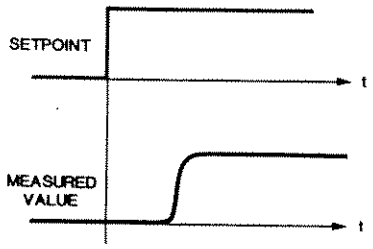

Fig. 29
Response curve with negligible time constant

If you find that the response curve of the open loop system appears to have no time constant at all but only a time delay, then you may need to actually build a time constant into your GEM 80 program. This may seem a bit odd, as it sounds as though we are trying to slow the system down. However, if you don't have any time constant, then theory says (see Page 29) that you can't have any loop gain M at all. If you try putting a closed loop round the plant, you are likely to get an oscillation at a frequency whose period is slightly more than twice the time delay. What happens is that the output signal from the GEM 80 will be driven too high, but the measured value doesn't respond for the time delay. When eventually it does respond, the output from the GEM 80 is now driven too low, and, again, the measured value doesn't respond for another time delay. The cycle then repeats.

To put in a closed loop system and make it stable, you need to artificially provide a time constant in your program. Since you can make this as big as you like, then you can also make M as big as you want. The fastest response you will then get from the closed loop system is where the effective time constant equals the effective time delay. This effective time constant will, of course, be a lot smaller than the time constant you have built into your program.

Instead of a time constant, you could use an integral term. We will come back to this later when we have learnt a bit more about integral terms.

CASE 3: TIME CONSTANT AND TIME DELAY ALMOST EQUAL

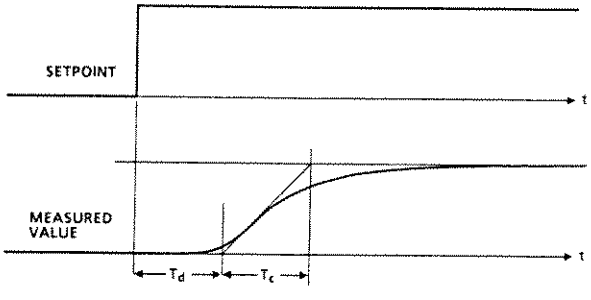

Fig. 30
Response curve with equal time constant and time delay

If your open loop response shows that the effective time constant and time delay are about equal, then our previous rules indicate that $M_{max}+1$ can only equal about 1, or M_{max} is about zero.

If both values are short, then you may be able to adopt the same approach as in Case 2 above, i.e. you introduce a much longer time constant into your GEM 80 program. The plant time constant, being small compared with this, can then effectively be treated as adding to the time delay.

However, if the open loop response is slow and quite S-shaped, with a very rounded start rather than having a pure time delay, then it is quite likely that your plant contains two time constants of similar size. In this case, you should study your plant to see whether you can identify two separate storage elements in it, each producing a time constant. The best approach in this case is usually to see whether you can use two separate control loops, each containing just a single plant time constant. This topic is covered later in Section 2 in the Item on "Cascade Controls".

CASE 4: OPEN LOOP RESPONSE OSCILLATORY OR UNSTABLE

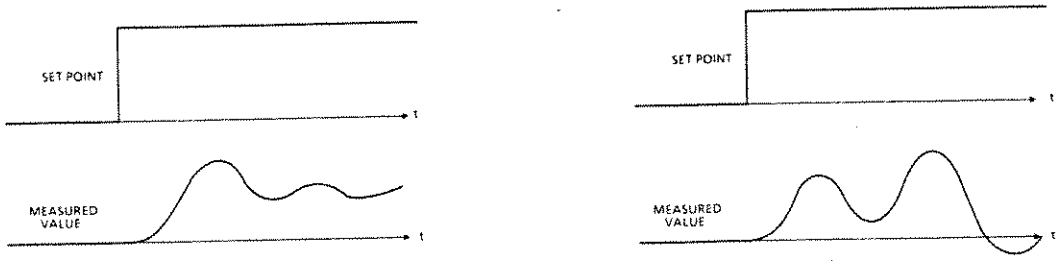

Fig. 31
Open loop response oscillatory or unstable

If your open loop response plot looks like either of the above, then your plant itself has some internal feedback mechanism acting inside it, and probably two or more time constants. We are then getting to the point where the system may need analysing properly, as standard rules-of-thumb may not necessarily work.

Possible ways in which a stable system with a good response may be achievable are:

- using a subsidiary transducer and a 2-loop cascade control (see Item in Section 2)

- using derivative (rate-of-change) feedback (see later in this Section).

But, which one (if either) of these approaches will work satisfactorily depends on the particular system, so you need to have a fair understanding of what is actually happening in your plant which causes the wobbly open loop response.

CASE 5: SYSTEM CONTAINING AN INTEGRAL

Some systems contain storage elements with no losses. For instance, the contents of a tank will continue to increase as long as there is any flow into it (assuming that there is no flow out of it and that the tank does not leak).

If you tried to control the liquid level in the tank open loop, you would not be able to use an open loop system as in Fig. 1. If you did, it is obvious that, for any value of setpoint, the level in the tank would carry on rising unless, after a certain time, you altered the setpoint back to zero. To set the level to a given value, you would have to arrange for a certain value of setpoint to be used for a defined time.

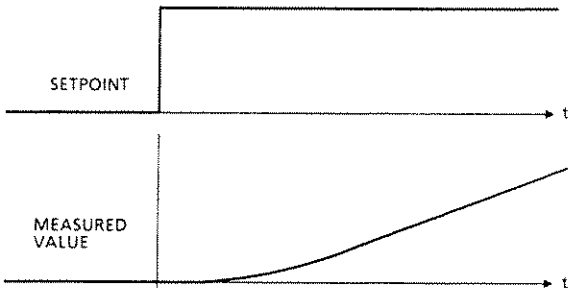

Fig. 32
Response curve for system with integral

Note that, to set the level to the same value, you could use twice the setpoint (to double the flow) but leave the setpoint switched on for half the time that it would have been previously. What is therefore important is the product of the flow and the time, or, if you like, the area under a flow-time graph. In mathematical parlance, the level in the tank depends on the integral of the setpoint with respect to time.

o **An integral term represents storage with no losses**

We said earlier that a time constant represented storage with losses, and that reducing the losses made the time constant longer. If we have no losses at all, the time constant becomes infinite, and we then have an integral term.

Coming back to our liquid level contol, we note that, if we set up an open loop system as in Fig. 1, what our setpoint actually controls is rate-of-change of level, not level itself. However, because we intend to use closed loop control, then we will need to make the setpoint span of values equal to the span of values we get back from our level transducer, in preparation for going closed loop (see under the heading "Setpoint Span", Page 20). Having arranged this, we can then take a recording in a similar way to Fig. 17, but starting with the setpoint giving zero flow into the tank, and switching it back to this zero-flow value before the level gets too high (see Fig. 32).

What we have to do, in the case of a system containing an integral, is to make an additional measurement from our recording. In Fig. 18, all we did was to measure times; with an integral, we also have to measure gains. We therefore need to actually measure the size of the setpoint change, and the corresponding output change. With a system containing only a time delay and a time constant, we didn't have to bother about the size of the setpoint change (as long as it was small) or the corresponding change in the output. The recording we might get for an open loop system containing an integral term could be as in Fig. 33.

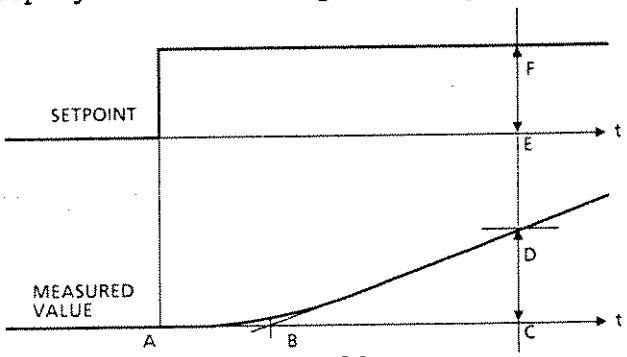

Fig. 33
Measuring Plant Integral Time and Time Delay

The measurements we need to take off this recording are:

1. Draw a straight line following the output ramp down to the zero output line, cutting it at B.

2. Mark off on the initial value line the time at which the setpoint changed, at A.

3. At some point sufficiently further on in time for the output to be ramping steadily (i.e. after any initial rounding has ceased), drop a vertical to the time axis, at C.

4. Measure the size of the output (C to D) and the size of the setpoint (E to F).

Then:

The effective plant time delay T_d is the time from A to B.
The effective plant integral time T_i is the time from B to C, multiplied by EF and divided by CD.

What control theory tells us is that, if the open loop system has a response consisting of a time delay and an integral term, then the closed loop response will consist of a time delay and a time constant. If the gain we use is M times larger than the gain we were using with open loop control, then:

- o **The effective system time delay is the same as that of the open loop system.**

- o **The effective system time constant equals the effective plant integral time divided by the factor M.**

- o **The highest gain factor M which can be used while maintaining a reasonably good, stable response, is that which makes the system time constant roughly equal to the system time delay.**

The second and third clauses tell us that the maximum gain factor M we can use while maintaining a good response is of the order of:

$$M_{max} = \frac{T_i}{T_d}$$

where T_i = effective plant integral time

T_d = effective plant time delay

However, with a system containing an integral term, accuracy depends only on the accuracy of our process measurement, in the steady state. We deal with the reasons for this in the next Item on "Proportional + Integral Controls".

----o0o----

INTRODUCTION

We have been concentrating in the last few items on various aspects of response and stability. However, we did start off by looking at how the accuracy of a control could be improved, despite disturbances, by using a closed loop. We found, however, that with a proportional control there was a limit to how high we could make the loop gain M because of stability. Since the effect of disturbances was reduced by the factor M+1 compared with an open loop system, stability therefore limited the steady state accuracy we could achieve.

However, we did mention in passing (Page 31) that we could improve the accuracy by adding in an integral term, turning the control from a proportional control into a proportional + integral control, or PI-control for short. This is the most commonly used form of 2-term control.

INTEGRAL TERMS

In the last Item, on "Special Cases", we came across integral terms in our "Case 5". We said that an integral term represented storage with no losses. In a GEM 80 program, the equivalent of an integral term is an up-down counter. In ladder diagram terms, the following rung is an integrator:

Fig. 34
GEM 80 rung for an integrator

What this rung does is to take the value held in table location G101 worked out on the previous execution of the program, add on to it the value held in G100, and put the answer back into G101 ready for the next execution of the program. If G101 initially contained zero, and we made G100 equal to 10, then on successive executions of the program G101 would take on the values 10, 20, 30, etc. If we now set G100 to zero, then, on subsequent executions of the program, G101 would retain whatever value it had reached. To reduce the value held in G101, we would have to make G100 contain a negative number, e.g. -10, and then G101 would count down again.

If we made G100 double what it was (e.g. 20), then the value of G101 would climb twice as fast. Starting with G101 equal to zero, the value of G101 at any time subsequently depends therefore on both the value of G100 and the number of program executions. If we have made the program such that its execution occurs at regular intervals (which is essential for analog controls: see comments under "Sampling Delays", Page 32), then the value of G101 is proportional to the area under a graph plotted of G100 against

time, or in other words the integral of G100 with respect to time.

You will see from the above description how similar the behaviour of the rung is to the level control we discussed in "Case 5" in the last Item.

GAIN WITH AN INTEGRAL

For the same input signal, we can obviously alter how fast the integral term is by putting in a gain before the ADD instruction in the rung:

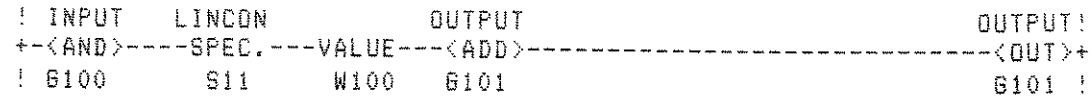

Fig. 35
GEM 80 rung for an integral with gain

Thus, if our LINCON has a gain of 2, and we make G100 equal to 10 with G101 initially zero as before, then G101 would take on values of 20, 40, 60, etc., i.e. G101's value would climb twice as fast as before.

USE OF INTEGRAL TERMS IN CLOSED LOOP CONTROL

The useful property of an integral term which we can make use of in a closed loop control system is that, for any input signal at all to the integrator (G100 in the previous two Figs.), the output will change. The output (G101) will only stop moving when the input is zero. From an integral, therefore, we can get an output with no input. The output does not depend just on the current value of the input, but on its past history as well.

So, our first bright idea might be to use a control rung for our closed loop control, equivalent to Fig. 12 but using an integral instead of a proportional gain, as below:

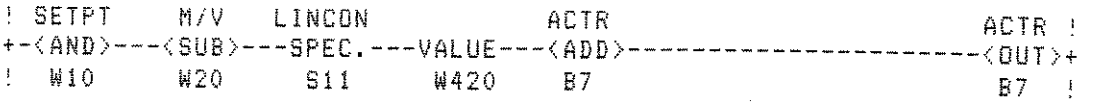

Fig. 36
First attempt at making an integral control

What (theoretically) ought to happen is that, when we change the setpoint on W10, we get a difference between W10 and W20 values, so that the output to our valve actuator from B7 starts to increase. As the flow starts to increase, then our measured value coming back from our flow transducer into W20 increases, reducing the difference towards zero. The signal from B7 only stops increasing when W20 becomes exactly equal to the setpoint W10.

So, what we have achieved by using an integral term is to get a control where the setpoint and the measured value under steady state conditions are exactly equal. To achieve the same result with a proportional control, we would need a loop gain M which was infinite. In fact, the rules for the accuracy of an integral control can be derived from those for a proportional control (Page 25) by setting M to infinity:

- **The effects of disturbances are reduced, in the steady state, to zero.**

- **If our open loop system has a non-linear characteristic between setpoint and output, the closed loop control will be completely linear.**

Compare these statements with the results of having a proportional control with a finite loop gain of M given on Page 25. Because we have no difference between the values of setpoint and measured value in the steady state, then we don't need any scaling of the setpoint value before we subtract the measured value, as we did in the proportional control rung of Fig. 15.

Because we have got the equivalent of an infinite loop gain, with the effects of disturbances reduced to zero, then the system accuracy depends solely on the accuracy of our measurement. (Note: This is only true for digital systems like GEM 80, where the measured value is subtracted from the setpoint in a numeric calculation. In analog systems, inaccuracies and drift may be introduced in the subtraction itself, e.g. in an op. amp. input circuit.)

Unfortunately, there's a snag. If the open loop response of the plant is of the typical form shown in Fig. 17, equivalent to a time delay and a time constant, the system is likely to be unstable. We may be able to stabilize it by making the LINCON have a very low gain, but we then finish up with a very slow response, maybe slower than the open loop response. So our first bright idea is not necessarily quite so bright after all.

PROPORTIONAL + INTEGRAL

So let's try combining the best features of proportional and integral controls. A proportional control can be used to give us a fast response, but does not reduce the difference between setpoint and measured value to zero, whereas an integral control reduces the difference to zero but can't be made very fast. In fact, we find that, if we include both integral and proportional terms, we can make the integral very much faster than using an integral on its own. We do, in fact, get the best of both worlds. The only minor drawback is that we have to drop the proportional gain slightly compared with what it would have been for a purely proportional control. We therefore finish up with a fractionally slower response compared with what could be achieved with a proportional-only control system. More on this in the next Item.

SUMMARY ON THE ACCURACY OF CLOSED LOOP CONTROLS CONTAINING INTEGRAL TERMS

(Compare with the Summary on the Accuracy of Proportional Closed Loop Controls given on Page 27; notice how the inclusion of an integral term simplifies the rules).

1. The total accuracy of the system in the steady state is equal to the accuracy of the measurement.

2. If the accuracy of measurement isn't adequate, you will have to choose better equipment for the measurement.

-----oOo-----

INTRODUCTION

In the GEM 80 armoury of Special Functions, there are functions specifically intended for 3-term control. However, we will begin by looking at how we might perform PI-control if we didn't have these Special Functions available, as it will give us a better insight into how these functions work.

PI-CONTROL USING DISCRETE FUNCTIONS

Fig. 37 below shows how we could make a proportional + integral control using separate rungs for the two terms, with a third rung to add them together:

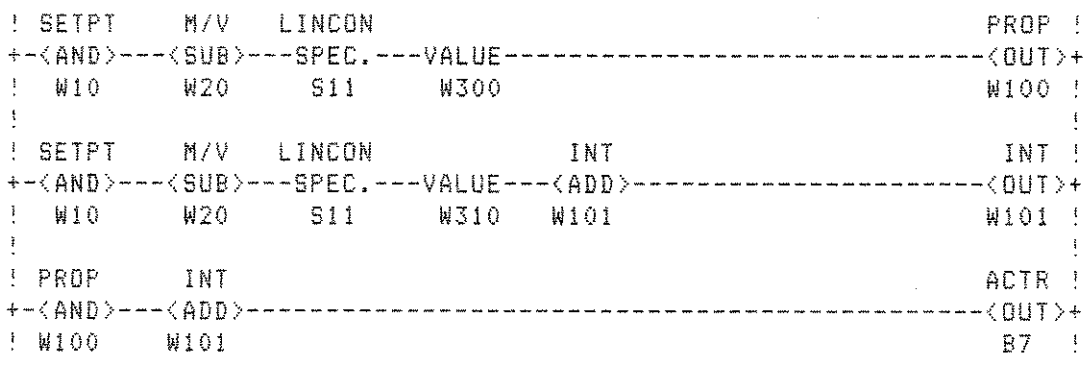

Fig. 37
PI-control using separate rungs

In the first rung, we produce an intermediate output in W100 proportional to the difference between the setpoint and measured values, i.e. to the difference between W10 and W20 values; the proportional gain is set by the LINCON in this first rung. Similarly, in the second rung we produce an intermediate output in W101 which depends on the integral of the difference between W10 and W20; the speed of the integral depends on the gain of the LINCON in this second rung. Finally, we add the two terms together in the third rung and use this for the output to the actuator.

Normally, the gain of the first LINCON will be much higher than that of the second LINCON; we will see why later. If we consider the system initially in a steady state condition with no difference between W10 and W20, then W100 will contain zero, but W101 will contain a value sufficient to set the output B7 to the required output value. If there is now a change in setpoint W10, or a change in measured value W20 due to a disturbance, then W100 will change a relatively large amount, whereas W101 will change a much smaller amount, due to the different LINCON gains. On the next program execution, however, W100 will not change any further, whereas W101 will change by a further small amount, and the same thing will happen on further program cycles. The resulting changes in output to the plant will tend to bring the measured

value towards the setpoint value, so that the difference between W10 and W20 tends towards zero. This will drop W100 also towards zero, and reduce the rate at which the value in W101 was climbing towards zero. Eventually a new steady state condition will be reached where W100 again contains zero and where W101 holds a value equal to the new output to B7. Diagrammatically, the two values W100 and W101 will add together to form B7 as below:

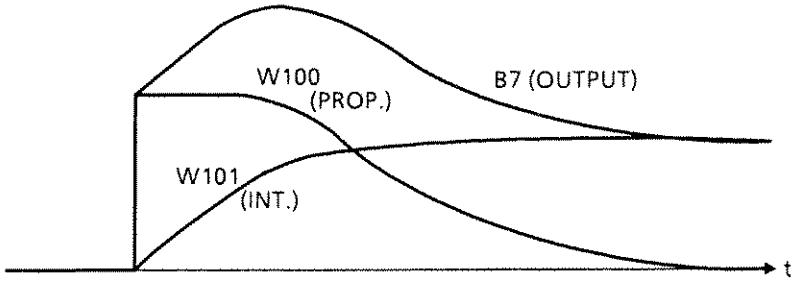

Fig. 38
Response of proportional and integral terms

PROPORTIONAL AND INTEGRAL GAIN SETTINGS

Proportional Gain

For a proportional control system, we only had one LINCON to bother about as regards stability. (True, if you look back at Fig. 15, we had two LINCONs, but only the second one affected stability; the first one was just to get the spans of the setpoint and measured value to match exactly, and its setting did not affect stability as it came outside the closed loop. This is clearer, maybe, from looking at the block diagram in Fig. 14). With a proportional + integral control as in Fig. 37, we have two LINCONs, both of which affect stability.

If we set the LINCON in the second rung to have zero gain (i.e. if we eliminate the integral term), then we would know how to set the first LINCON since we would just have a proportional control. We summarized the rules on Page 37 at the end of Item 5 on "Proportional Control Stability". You will remember that we could set the gain M times larger than what we would have used with open loop control for the same setpoint span, and that M+1 had to be no larger than the ratio of the effective plant time constant to the effective plant time delay.

If we are using PI-control, then adding on the integral term is going to make the system fractionally less stable. We must therefore drop the proportional gain slightly to make the system a bit more stable to start with, in preparation for adding on the integral term. Most people recommend reducing the proportional gain to 90%. This, then, gives us a rule for how to set up the gain of the LINCON in the first rung of Fig. 37.

Integral Gain

Now let us consider the second LINCON, the one setting the speed of

the integral term. We have already considered in the previous Item the way in which the output from an integral ramps up when it has any input, and how it only stops ramping when the input to it is reduced to zero. Suppose that there were a 1-bit difference between W10 and W20 in the second rung of Fig. 37. Then the rate at which W101 would ramp up would not just depend on the setting of the LINCON in this rung: it would also depend on the time interval between executions of this program rung. Suppose that our LINCON had a gain of 5; then the ramp rate of the integral term would obviously be quite different with a short execution interval and a long execution interval, as illustrated in Fig. 39 below:

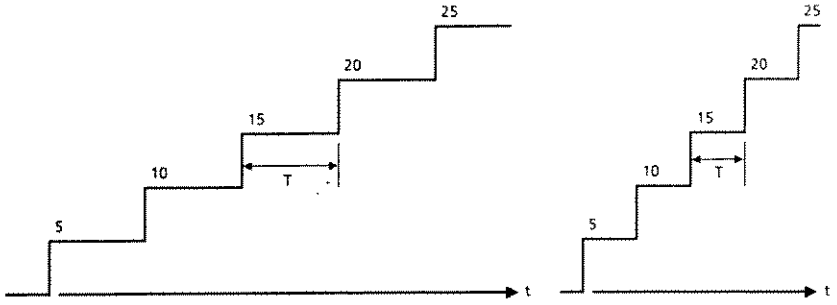

Fig. 39
Effect of program repetition interval on integral terms

The gain setting for our LINCON is therefore dependent on the frequency of execution of the rung. That is, if we make the treads of the stairs wider, then we need higher risers to keep the slope of the staircase the same. To get a certain ramp rate for a 1-bit error, the gain will be proportional to the execution interval, which we will call T (without any subscript).

Note (as we have remarked before) that T must be constant. If our integral rung were executed at irregular intervals, then we wouldn't get a uniform staircase, and the performance of our control system would not be predictable.

If we have set the LINCON in the first rung of Fig. 37 (the one setting the proportional gain) to a gain of k_p giving a loop gain of M, then the rule-of-thumb formula for the gain of the LINCON in the second rung (the one setting the integral rate) is:

$$k_i < 0.3 \frac{T}{T_c} M k_p$$

Now, if we have set the proportional gain LINCON to give a loop gain of $0.9 M_{max}$ (as given on the previous page), then assuming that M_{max} is large compared with unity, we can say that T_c/M_{max} is approximately equal to the effective system time delay T_d, giving the corresponding maximum gain for the second LINCON of:

$$k_{i_{max}} = 0.3 \frac{T}{T_d} k_{p_{max}}$$

We said earlier that, to prevent increasing T_d too much, we should not make our sampling interval more than about 20 to 25% of the plant dead time, i.e. our BLOCK repetition interval T should not be more than about 0.2 T_d. Hence, from the second formula, we can expect k_i to be around 6% of k_p as a maximum value. (This verifies a statement we made in the last paragraph on Page 54, that the gain of the first LINCON will normally be much higher than that of the second LINCON.)

Note that, if we double k_p, then for a similar but twice-as-fast response, we can quadruple k_i. This is because, for any loop gain M less than M_{max}, k_p is itself proportional to M, so we have got a square law for the permissible integral gain k_i against k_p. If we were setting the control up entirely by cut-and-try methods closed loop (without first taking a recording of the open loop response), starting with low gains for our LINCONs but then winding them up, we would therefore be able to increase the LINCON for the integral term by a larger factor than that for the proportional term.

RESPONSE

In theory, as long as you've got an integral term, it doesn't matter what its gain is when you consider steady state accuracy. However, if you alter the setpoint, or if there is a disturbance to the plant, the measured value will be pulled back quicker to again equal the setpoint if you can make the integral gain higher. This is why the formulae on Page 56 tell you what the maximum gain can be with a reasonable response.

This idea of the integral term causing the measured value to be brought back into equality with the setpoint gives rise to another piece of process control engineers' terminology - they refer to a controller with an integral term as having "reset" action. In other words, such a controller resets the measured value equal to the setpoint.

DESIGN EXAMPLE

Suppose we start off running our system open loop so that we can measure its response as in Fig. 18. To do this, we must first edit the rungs in Fig. 37 as below:

```
! SETPT   LINCON                                                  PROP !
+-(AND)---SPEC.---VALUE-----------------------------------------(OUT)+
!  W10     S11     W300                                          W100 !
!                                                                     !
! SETPT    M/V   LINCON(zero gain) INT                            INT !
+-(AND)---(SUB)---SPEC.---VALUE---(ADD)----------------------(OUT)+
!  W10     W20     S11     W310   W101                          W101 !
!                                                                     !
! PROP     INT                                                   ACTR !
+-(AND)---(ADD)-----------------------------------------------(OUT)+
! W100    W101                                                   B7   !
```

Fig. 40
PI-control with rungs edited for open loop running

What we have done here is to temporarily remove the <SUB> W20 from the first rung, and we haven't put any data in the VALUE tables for the second LINCON so that, by default, its gain is zero. (Note that it will produce a fault code in W310 for divide-by-zero, but this is of no consequence.) We also set the contents of W101 to zero (if it isn't already). Because we have set the LINCON gain in the second rung to zero, W101 will then remain zero regardless of any changes in W10 and W20. As a result, the output signal from B7 to our actuator will just be equal to W100.

Suppose we want our setpoint in W10 and measured value in W20 both to span the range 0 to 10000. For open loop running, we need to set the gain and offset of the first LINCON so that, as we vary the setpoint in W10 over this range, we get the correct span of values out of B7.

For our example, we will assume that we are driving a 4 to 20 mA signal from an analog output module which needs a value of 2000 to give 20 mA, and a value of 400 to give 4 mA. We might find from experiment, however, that because of end effects, the practical working range for our actuator was only from 5 mA to 19 mA. This would require values to B7 of 500 and 1900 respectively.

Therefore B7 needs an input which has:

Offset = 500
Span = 1400

To get these values when W10 ranges from 0 to 10000, we need our LINCON to have:

Offset = 500
Gain = 1400/10000

We could therefore set our table values to:

W301 = 1400
W302 = 10000
W303 = 500

Suppose we now conduct an open loop response test. We can run the plant somewhere in the middle of its range by setting W10 to 5000, say, which will give an output from B7, also in the middle of its range, of 1200. If we now increase W10 from 5000 to 5200, say, we can record the response of the measured value W20, and measure the effective system time delay and time constant as in Fig. 18.

In practice, it may be difficult to record W20 itself, because this is a table value internal to the GEM 80 controller. We will deal with some of the possible practical ways of making recordings in Section 4. However, let us assume that what we measure from a recording is:

time delay = 10 seconds
time constant = 140 seconds

In this case, these times are very long compared with the program cycle time we can expect to get from a GEM 80 controller. Remembering what we said about sampling delays (Page 33f), we can therefore put the control in a BLOCK obeyed every 2 seconds. This will give us an additional delay of half the BLOCK execution interval, or 1 s, giving us 11 s total. The controller time delay will be very short in comparison with this, so we can ignore it. The effective system time delay T_d will therefore be 11 s in total.

For the proportional gain, we can increase the gain of the LINCON in the first rung from its open loop gain of 1400/10000 up to a maximum of M times this, where:

$$M+1 = \frac{T_c}{T_d}$$

$$= \frac{140}{11}$$

$$= 12.72$$

or: $M = 11.72$

To achieve this, we could increase the LINCON numerator to 1400 x 11.72, or 16408 to the nearest integer, if we were using just proportional control. However, if we are going to add on an integral term, we should set the proportional gain to only 90%, or the numerator should be 16408 x 0.9 = 14767.

Since we have set the proportional gain to the maximum recommended value, then the gain for the second LINCON giving the integral rate will be:

$$k_i = 0.3 \times \frac{T}{T_d} \times \frac{14767}{10000}$$

where: T = block execution interval = 2s
T_d = effective system time delay = 11s

so that $k_i = \frac{805}{10000}$

We can, then, enclose our three rungs in a BLOCK obeyed at 2 second intervals and set:

```
W301=16887
W302=10000
W311=921
W312=10000
```

We then reinstate the <SUB> W20 in the first rung; and hopefully our system will perform as anticipated.

Note that, once we have got the integral term operative, there is no necessity for keeping the offset (W303=500) which we put in for open loop running, as the integral term will always bring the

output to the correct value, steady state, with zero difference between the setpoint and measured value.

SUMMARY ON THE RESPONSE OF PROPORTIONAL + INTEGRAL CONTROLS

1. Work out the gain of the proportional term first, as though the control had no integral term, following the rules given previously.

2. For the fastest response, set the proportional gain to 90% of the maximum theoretical gain, to allow for the destabilizing effect of the integral term.

3. The integral gain is proportional to the execution interval. If the control rungs are in a BLOCK, this interval will be the BLOCK repetition interval; if they are not in a BLOCK, this interval will be the program cycle time.

4. The formulae given on Page 56 give the maximum integral gain which can be used when the proportional term would, on its own, have a loop gain of M, and also the maximum integral gain which can be used when the loop gain is set to 90% of M_{max} (see 2. above), with a reasonably stable response.

5. The value of integral gain makes no difference to the steady state accuracy, but a higher gain will make the measured value reach a value equal to the setpoint quicker after a disturbance or setpoint change.

----o0o----

INTRODUCTION

Producing a PI-control with three rungs as shown in Fig. 37 may not look particularly complicated. We can, however, do the same with just one rung using the PIDABS Special Function S34, as shown below:

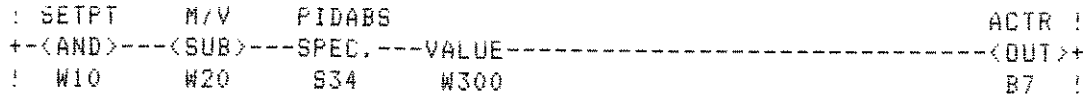

Fig. 41
PI-control using PIDABS Special Function

However, the PIDABS Special Function, besides being simpler to program, also gets round some problems we haven't so far considered, the chief of which is "integral wind-up".

THE NEED FOR LIMITS

All closed loop control systems can be expected to behave respectably while all the signals are somewhere in the middle of their span. But nasty effects can happen when signals get to either end of their span of values.

Suppose we have a large change in setpoint. Our PI-control will then cause a large change in output signal. This could be larger than the maximum our valve actuator or other device controlling the plant can respond to. Once we reach the stage where any further change in control signal to the actuator will be ignored, we have lost our closed loop control action, and peculiar things can be expected to happen (and generally do!).

In fact, you can take it as a piece of wisdom that half the problems with closed loop control systems have nothing to do with the operation of the closed loop itself, but with:

- how you start the system up to get it into closed loop mode.
- how you take it out of closed loop mode (e.g. on auto-manual change-over).

In our case, if the signal from our GEM 80 to the plant gets sufficiently large for the plant to ignore any further increase in signal, our closed loop control doesn't "know" why the plant is no longer responding. It will just carry on making the signal bigger. Eventually, as the measured value increases closer to the setpoint, the signal to the plant will start to reduce. It might, however, have to come down a long way before it was small enough for the actuator to start responding again. By that time, we could have had a large overshoot of the measured value, as shown in Fig. 42.

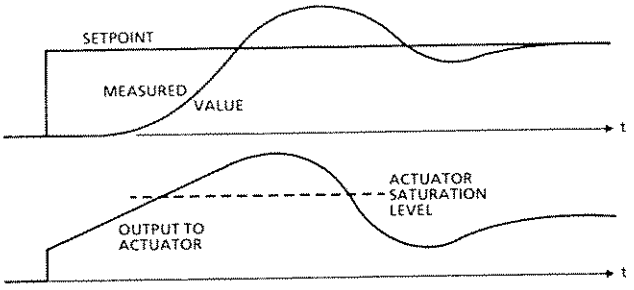

Fig. 42
Overshoot caused by actuator saturation

The overshoot problem therefore arises if our controller provides signals which exceed the input span of the actuator or other controlling device. On the face of it, the problem is easily overcome by using the LIMIT Special Function S32. You could modify the last rung of Fig. 37 to become:

```
! SETPT     M/V    LINCON                                          PROP  !
+-(AND)---(SUB)---SPEC.---VALUE-------------------------------(OUT)+
!  W10      W20      S11      W300                                W100  !
!                                                                      !
! SETPT     M/V    LINCON              INT                          INT  !
+-(AND)---(SUB)---SPEC.---VALUE---(ADD)-----------------------(OUT)+
!  W10      W20      S11      W310    W101                        W101  !
!                                                                      !
! PROP      INT     LIMIT                                          ACTR  !
+-(AND)---(ADD)---SPEC.---VALUE-------------------------------(OUT)+
! W100     W101      S32      W350                                 B7   !
```

Fig. 43
Output rung with output limit

Note that the system while in limit is not operating in closed loop mode, but is operating open loop with a fixed value of signal (i.e. the limit setting) as output to B7. Compared with the situation when we had no limits, however, it is our program which decides what is going on; we are no longer being "dictated to" by the plant as to when we are operating closed loop and when we are operating open loop.

INTEGRAL WIND UP

Unfortunately, this "solution" produces a nasty. While the difference between the setpoint and measured value is non-zero, the second rung carries on counting up, so that W101 could reach quite a large value. The only way in which W101 can come back down again is for the difference to reverse, i.e. the measured value must become larger than the setpoint, for quite some time. We finish up with a much larger and longer overshoot than we would normally expect to get for such a control. Fig. 44 shows how the proportional term in W100 and the integral term in W101 might behave, as well as the measured value in W20, for small changes and for large changes of setpoint.

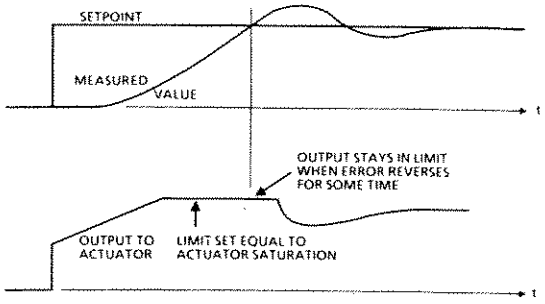

Fig. 44
Response without integral wind-up prevention

Our next idea might be to put a LIMIT Special Function in the second rung as well. However, since in the steady state the setpoint is exactly equal to the measured value, then W100 will contain zero, and the whole of the output to B7 must come from the W101 integral term. We therefore can't set a LIMIT in the second rung to any lower value than the LIMIT in the last rung. We might then get responses slightly better for large setpoint changes than shown in Fig. 44, but we haven't totally cured the problem.

Further, our control rungs are beginning to get rather complicated. At this stage, then, we'll abandon trying to do 2 or 3-term control using individual rungs with discrete functions, and concentrate on using the PIDABS Special Function as in Fig. 41, which has inherently got these problems buttoned up.

RATE LIMITS

Besides limits for the minimum and maximum number output, PIDABS also contains rate limits. These limit how rapidly the output can increase or decrease, i.e. what the maximum increase or decrease in output number per program execution is permitted to be (rather like a RAMP Special Function). You should make use of these where the actuator or other controlling device you are providing an output to from the GEM 80 cannot operate faster than a certain rate, even if you change the output rapidly.

In other words, all limits should be set to match the characteristics of the actuator so that you can never provide an output signal larger or changing faster than the actuator is capable of responding to.

SUMMARY ON LIMITS

1. Set limits and rate limits so that you provide a signal to the plant which is never larger or changing faster than the actuator or other controlling device on the plant can respond to.

2. Wherever possible, use the limit facilities built into the Special Functions provided for closed loop control in your GEM 80 controller.

3. Where you have a number of functions in cascade, don't just rely on limits on the final function in the chain. Check first that signals from earlier functions in the chain can't wind up.

----oOo----

INTRODUCTION

We have now covered 2-term PI control. The last term which can be added is a derivative term, giving us 3-term PID control.

A derivative term only gives an output when its input is changing. It has no effect on accuracy, in the steady state. As long as our process measurement is accurate enough to meet our requirements, then an integral term ensures that the setpoint and measured value become exactly equal in the steady state.

However, integral terms are too slow to correct for rapidly changing disturbances. With a 2-term control, we rely on the proportional term to provide rapid corrections. We showed that the faster we could make the response of a proportional control, the quicker we could bring the measured value back into line with the setpoint (see Item 7 on "Proportional Control Response to Disturbances").

What a derivative term does is to further improve the response to rapid disturbances. Also, a derivative term tends to make a control somewhat more stable, in contrast to an integral term which tends to make a control less stable. When a derivative term is added, we can normally set the proportional gain higher, and increase the integral rate as well.

DERIVATIVE TERM USING SEPARATE RUNGS

To see how a derivative term works, we will first consider how we could construct a derivative term from discrete elements, in a similar way to how we considered integral terms.

For a derivative term, we only want to give an output when the input changes. This means that we have got to remember the last value of the input, and subtract it from the present value of input to find any change. This can be done with a couple of rungs as below:

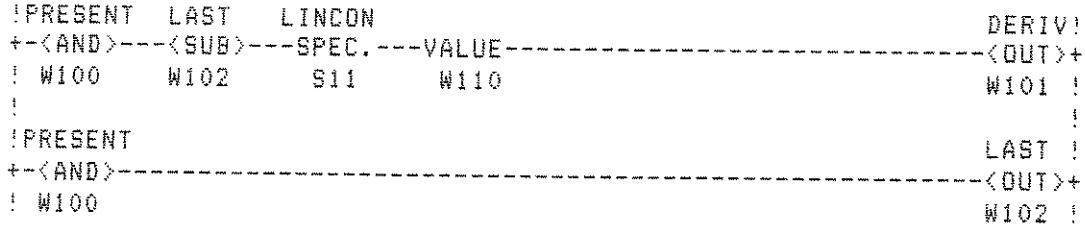

Fig. 45
Derivative term using separate rungs

The first of the two rungs subtracts the last value of input from the present value of input. If the input has not changed since the last program execution, there will be no output from this rung. However, if the input has changed, then the difference is

scaled up by the LINCON depending on the required gain to provide the output W101. The second rung then updates the last value in W102, ready for the next execution of the program.

Fig. 46 below shows the way in which the output from a derivative term on its own, like W101 output in Fig. 45, responds to step changes and ramp changes on its input:

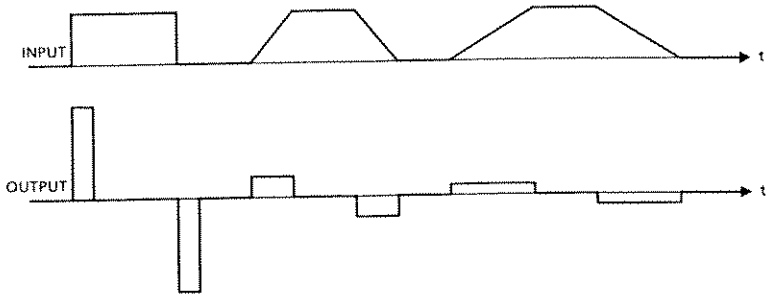

Fig. 46
Response of a derivative term

Note particularly the case of the ramp input. The output from the derivative term is proportional to the slope of the ramp, i.e. the output is proportional to the rate-of-change of the input, or, in mathematical parlance, to the derivative with respect to time of the input.

LIMITATIONS OF DERIVATIVE TERMS

If derivative terms are a good thing, tending to make systems more stable and speeding up their response, then why don't we use them all the time, and why don't we make their gain extremely high?

The first reason is because derivative terms tend to amplify noise. We showed in Fig. 21 how the input to a GEM 80 could fluctuate when sampling a noisy signal. Because this signal can vary on every program cycle, then the output of a derivative term, given this sort of input, will swing positive and negative. If the LINCON in the first rung of Fig. 45 is set to a gain higher than 0.5, then the peak-to-peak amplitude of the derivative output will be bigger than the amplitude of the ripple on the input.

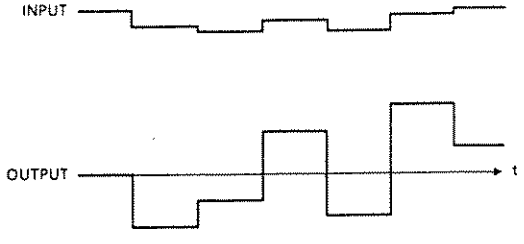

Fig. 47
Output of a derivative term with a noisy input

(Note: With a GEM 80 or other type of programmable controller, the effect of sampling eliminates high frequency noise. This doesn't happen with a purely analog controller. Since the output of a derivative term is proportional to the rate of change of the input, then high frequency noise, even of quite low amplitude, could get amplified to an unacceptable level. Consequently, analog controllers always have some band stop in their circuits so that derivative action ceases above a certain frequency.)

The second reason for not having too much derivative action is that, although a certain amount of derivative action does make a system more stable, too much derivative action can cause it to go unstable at half sampling frequency, where the output swings too high and too low on alternate program executions.

Consequently, for the majority of systems, the best you can do with derivative action is to increase the speed of response by somewhere about 30% compared with what it was without derivative action.

GAIN SETTING OF A DERIVATIVE TERM

Although, in a practical system, you are more likely to be using a PIDABS Special Function S34, let us consider how we might set up the derivative term given by the couple of rungs in Fig. 45.

For a step change of input, we get an output from W101 equal to the size of the change multiplied by the gain of the LINCON function. If W100 changes as a step, this output lasts for one program execution only, because the value in W100 remains constant after the step change. The overall effect on our plant depends on the size of the output and on its duration, i.e. on the area of the output when plotted as a graph against time. That is, you can either give the system a kick for a given period of time, or you can give it a bigger kick for a shorter time. The effect on the plant of the derivative term will be virtually unchanged if you double the gain of the LINCON but at the same time halve the program repetition interval T.

Because of this, we can expect the formula for the derivative gain k_d to depend on the reciprocal of the execution interval T. That is, for a given derivative action, the shorter we make T, the larger we need to make the gain of the LINCON in Fig. 45. Derivative gains therefore work back to front compared with integral gains, which are directly (not inversely) proportional to T (see formulae on Page 55).

Now consider a 3-term control. Suppose that we initially set it up as a proportional control without any integral or derivative terms. Then, whereas we had to reduce the proportional gain to 90% for a 2-term control, we find that with a 3-term control we can increase it to 120%. Also, compared with a 2-term control, we can increase the integral gain to 160%.

The formula for the integral gain becomes:

$$k_{i_{max}} = 0.5 \frac{T}{T_d} k_{p_{max}}$$

(Look back to Page 56 for the corresponding formula for 2-term control. It is the same, except that the constant is 0.3 instead if 0.5.)

Lastly, we have our third term, the derivative one. The formula for the derivative gain is:

$$k_{d_{max}} = 2.0 \frac{T_d}{T} k_{p_{max}}$$

As we expected, this gain is inversely proportional to the repetition interval T.

SPECIAL CASES

In Item 8 on "Special Cases", we mentioned that, for what we called "Case 4: Open loop response oscillatory or unstable", derivative action would sometimes help to get a good shape response. In such cases, the amount of derivative action we need can be very large. This can immediately get us into a noise problem because, as we noted above, the higher we make the derivative gain, the more we amplify any noise or ripple present on the measured value signal. If you can't reduce the noise by any of the methods given in Item 6, then you may need to use the ANALAG Special Function S37 to reduce the fluctuations. You mustn't, however, put in too much smoothing with an ANALAG, as you could then nullify the derivative action you are after.

In a similar way, you may also be able to get a better response for the type of "Case 3" system where the response is quite S-shaped.

For either type of system of the above types, however, you are better off if you can analyse the system mathematically - you are reaching the limits of what can be dealt with by the sort of rules of thumb given in this book.

SUMMARY ON THE RESPONSE OF PID CONTROLS

1. Work out the gain of the proportional term first, as though the control had no integral term, following the rules given previously.

2. For the fastest response, set the proportional gain to 120% of the maximum theoretical gain, to allow for the improvement in stability given by the derivative term.

3. The integral gain is proportional to the execution interval. If the control rungs are in a BLOCK, this interval will be

the BLOCK repetition interval; if they are not in a BLOCK, this interval will be the program cycle time.

4. The formula given on the previous page gives the maximum integral gain which can be used when the loop gain is set to 120% of M_{max} (see 2. above), with a reasonably stable response, when a derivative term is added as below.

5. The derivative gain is inversely proportional to the execution interval. See 3. above.

6. The formula given on the previous page gives the maximum derivative gain which can be used when the proportional and integral terms are set as given above.

----o0o----

INTRODUCTION

With any digital closed loop system, we generally find that the measured value is liable to continuously dither up and down by 1 or more bits, at a frequency which is quite low. In fact, if you didn't know about dither, you might blame this oscillation onto noise, or onto drift on the ADC converting the transducer signal into a number.

In this Item, we will therefore look at what causes dither, and how to get rid of it.

DITHER

What causes dither is a combination of:

- The resolution of the ADC on the measured value.
- The plant and PID controller gains.

(In a very few cases, it could depend on the resolution of the DAC providing the output to the plant as well, but this is rare. Any analog output module contains a DAC (digital-to-analog convertor). This is similar to the ADC (analog-to-digital convertor) used on analog input modules. A DAC has a characteristic like the staircase given in Fig. 7 for an ADC, except with the axes interchanged. Most DAC's will accept a 12-bit number for their input, and can produce 4096 discrete levels of analog output signal).

Now, it does not matter whether your controller works in integer or floating point arithmetic; what matters is that ADC's and DAC's only work in integer.

DITHER IN A PID-CONTROL

If you monitor the output from your GEM 80 when you change the setpoint, you may find that it behaves as in the Fig. below:

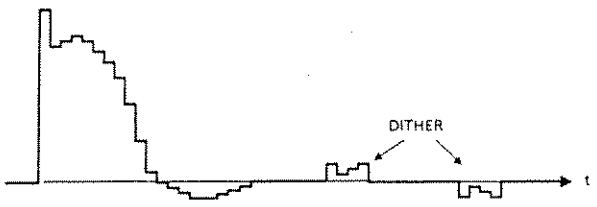

Fig. 48
Output dither from PID control.

Initially, the response behaves as you would expect it to, as we have described in previous Items. It gives one overshoot, and then settles down to a condition where the error is zero (as we

would expect for a control with an integral term), and the output from the GEM 80 to the plant remains constant.

But, after some time, the measured value changes by 1 or more bits. Since the error is no longer zero, we get a change in output to the plant as in Fig. 48. Then, after a short time, the measured value changes back to where it was, and everything seems steady again. But then, some time later, the measured value changes in the opposite direction by 1 or more bits.

To understand what is happening, let's look at a block diagram of the controller and the plant. We will assume that the ADC converting the trandsducer signal into a number gives an output which moves in 1 bit steps.

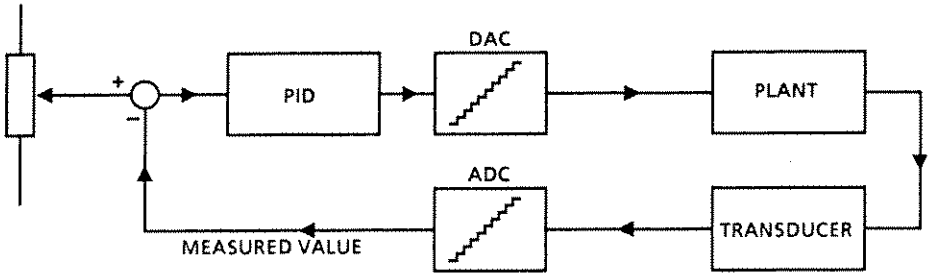

Fig. 49
Analog output to and input from the plant

What is happening is as follows. After the initial response, we get zero error, but the output to the plant finishes up 1 or more bits away from where it ought to be. As a result, although the transducer signal we are measuring was the correct value when we first had zero error, it carries on rising (or falling) very slowly (because of the plant time constant, and because of the incorrect output from the controller).

After some time, it moves sufficiently for the ADC to change its output by 1 bit. We now have a 1 bit difference between the setpoint and the measured value. This causes a change in the output to the plant. Because of the plant time delay, the plant does not respond until after this delay time. Once it does, the transducer signal drops off sufficiently for the ADC output to change back to its previous value.

We are now back in the condition where the error between the setpoint and measured value is zero again. However, the integral term will have counted up while the error was 1 bit. The output to the plant will therefore not now be the same as it was before. How much it will have moved depends on:

- The integral gain.
- How many program executions take place during a time equal to the plant time delay.

For most plants, the amount by which the output to the plant will have moved will be sufficient to make it now too large, if it were previously too small, and vice versa. Consequently, we get a continuous dither oscillation.

For a PI or PID control, the proportional and derivative terms make little difference, and it is the integral term which causes the problem. To know whether we are going to get a dither oscillation as described above, we must work out the value:

$$X = k_i \times e \times \frac{T_d}{T} \times k_{PL}$$

where:

k_i = integral gain

e = minimum non-zero error

T_d = system time delay (= plant + controller time delays)

T = repetition interval at which the PI or PID control is executed

k_{PL} = gain of the plant

By the minimum non-zero error e, we mean the smallest difference we can have between the setpoint and the measured value. This is not necessarily 1 bit. For instance, if we have scaled up the measured value, then a change of 1 step in the output number from the ADC may cause a change of several bits in the measured value, and we must then use this figure for the minimum non-zero error.

Given a change in input to the PI or PID control of e, we now get an output to the plant which counts up by $k_i e$ per program execution. Since the plant can't respond till after the system time delay, then the output to the plant counts up on T_d/T program executions. The change in output to the plant is therefore $k_i e T_d/T$. If finally we multiply this by the plant gain, we get the change we will get on the transducer signal.

The question now is: Will this change cause the ADC output to change by more than 1 step? If it doesn't, then you won't get any dither. If, as is more likely, it does, then your system will dither.

DITHER IN A PROPORTIONAL CONTROL

If our system does not contain an integral, we are still likely to get dither, but with an oscillation frequency which is somewhat faster.

In this case, the output to the plant can only change in steps of k_p, where k_p is the proportional gain. It is therefore quite unlikely that we can get the correct value of output to the plant

to make the measured value equal to the setpoint, but this will depend on the precise value of the setpoint.

Even if we can in theory get a correct value of output to the plant to make the measured value match the setpoint, it is still unlikely that it will settle down at this value. It is more likely to dither, just as with an integral. To know whether we are going to get a dither oscillation, we must work out the value:

$$Y = k_p \times e \times k_{PL}$$

where:

k_p = proportional gain

e = minimum non-zero error

k_{PL} = gain of the plant

The question again is: Will this change cause the ADC output to change by more than 1 step? If it doesn't, then you won't get any dither. If, as is much more likely, it does, then your system will dither.

HOW TO AVOID DITHER

The normal method of preventing dither is to put in a dead band on the error. That is, we arrange for the control to ignore minor differences between the setpoint and the measured value, and only get it to respond to significant errors.

Controls with Integral Terms

You can use the DEDBAND Special Function S31 in your control rung before the PIDABS as below:

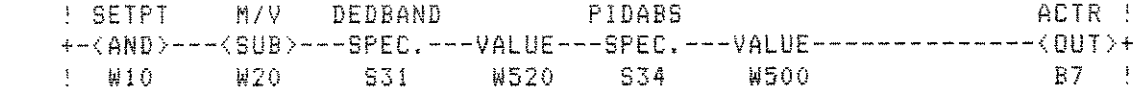

Fig. 50
Control rung including a dead band before PIDABS

(Note: It is most important that you put the DEDBAND before the PIDABS. Putting it after the PIDABS does not have the same effect, and doesn't work.)

For a control with an integral term, if you want to be certain of no dither oscillation, you need to make the size of the dead band at least $\pm X$ times the change you will get on the measured value for a 1 step change in ADC output (where X is worked out as on the previous page).

If you make the dead band smaller than this, you may find that the error can take you right across the dead band. You may then get a dither oscillation just as big as with no dead band, although it might decay to zero with a bit of luck.

With the DEDBAND Special Function S31, don't put any steps either side of the dead band. Set it up as at A in Fig. 51 below, and not as in B.

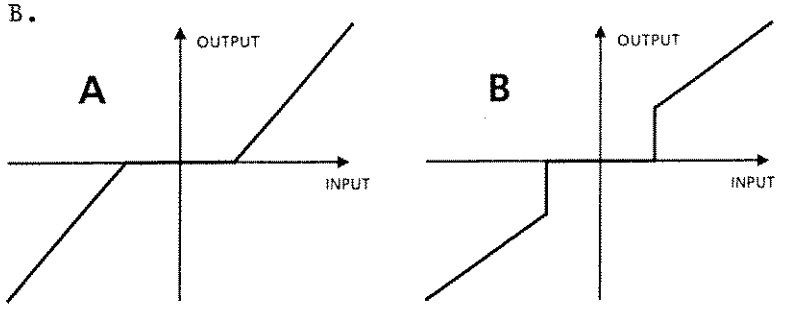

Fig. 51
DEDBAND Special Function

With a DEDBAND set to ±X times the result of 1 ADC step, we get a control which won't dither continuously, but it may dither a few times initially. To avoid this, you can set the dead band to ±Y/2, which is generally a wider dead band than ±X.

Controls with No Integral Term

If your plant contains an integral term itself (Special Case 5, Item 8), then in the steady state the difference between the setpoint and the measured value will tend to zero, and you can use a dead band. In this case, the width of the dead band should be ±Y/2 times the minimum non-zero error (where Y is worked out as on the previous page).

If, however, your plant contains no integral, then there will be some difference under steady state conditions between the setpoint and the measured value (see Item 4, Page 23). You can't therefore use a dead band unless you make it wide enough to allow for the variation in this difference, which could make it rather large. However, you can move the position of the dead band as in Fig. 52.

```
! SETPT     M/V     SHIFT  DEDBAND          SHIFT                DIFF !
+-<AND>---<SUB>---<SUB>---SPEC.---VALUE---<ADD>-------------<OUT>+
!  W10      W20     G101     S31     W520   G101                 G100 !
!                                                                    !
! DIFF    PIDABS                                                ACTR !
+-<AND>---SPEC.---VALUE-----------------------------------------<OUT>+
! G100      S34     W500                                          B7  !
!                                                                    !
! DIFF                                                         SHIFT!
+-<AND>-------------------------------------------------------<OUT>+
! G100                                                         G101 !
```

Fig. 52
Proportional Control with moving dead band

In Fig. 52, the top rung works out the difference between the setpoint and the measured value, but, before putting this into the DEDBAND Special Function, first subtracts the quantity called SHIFT. After the DEDBAND, it adds on SHIFT again. The output of this first rung is then fed into the 3-term control in the second rung, which gives the output to the actuator or other controlling device on the plant. Finally, the third rung sets up the value of SHIFT ready for the next execution of the program.

If the shifted difference between the setpoint and measured value comes within the dead band, then there will be no output from the DEDBAND Special Function, and the rung output DIFF will therefore just be equal to SHIFT. If the shifted difference exceeds the dead band, then DIFF will equal SHIFT plus the DEDBAND output. The third rung then alters the value of SHIFT ready for the next program execution, thus moving the effective dead band.

This technique allows you to set up the size of dead band to $\pm Y/2$, as for the case where the plant contained an integral.

EFFECT OF DEAD BANDS ON ACCURACY

Although dead bands prevent dither, they also reduce the system accuracy. Instead of the measured value and the setpoint becoming exactly equal on a control with an integral term, they can differ by up to the size of the dead band.

Note that no improvement would be made by making the program execution more frequent. In the formula for X (Page 72), if you reduced T you would also have to reduce k_i proportionately, giving exactly the same value for X, and therefore the same size of dead band.

Now, it all depends on your particular control system as to what you can accept. If you put in no dead band, you get dither, which may be acceptable on some controls. If you can't accept dither and do put in a dead band, then you lose out on accuracy.

There are, however, techniques for modifying the proportional and integral gains when the error is small, so that you don't get any difference between the setpoint and the measured value and yet get no dither. These techniques are a bit more complicated to program, and are dealt with in Section 2.

SUMMARY ON DITHER OSCILLATIONS

1. Dither oscillations may occur in a digital system because of the resolution of the ADC on the measured value in combination with the gains of the plant and of the controller.

2. Dither can be avoided by using a dead band operating on the difference between the setpoint and the measured value. With a proportional control, a smaller dead band can be used by shifting it.

3. If you use a dead band, then you will sacrifice accuracy. On a control containing an integral term, then, instead of the measured value becoming exactly equal to the setpoint, it can differ by up to the size of the dead band.

4. Methods of avoiding dither without dead bands, and which do not therefore lose out on accuracy, are given in Section 2.

----oOo----

INTRODUCTION

We looked at a few Special Cases in Item 8, and said that we would look further at one or two of them again later.

CASE 2 AGAIN: NEGLIGIBLE TIME CONSTANT

We said that, if your plant hadn't got a significant time constant, then you needed to put a large time constant into your GEM 80 program. We did, however, mention that you could also stabilize such a system using an integral term instead.

If your open loop controller gain is k_p, then you can use an integral term with gain:

$$k_i = \frac{T}{T_d} k_p$$

where:

T_d is the system time delay.
T is the execution interval of the integral term.

You can, of course, use a lower integral gain than this, but an integral gain which is not much higher than this will cause oscillations. You can use the PIDABS Special Function for the integral term, with the proportional and derivative terms omitted (i.e. with gains set to zero).

CASE 5 AGAIN: SYSTEM CONTAINING AN INTEGRAL

At the time we were dealing with this case in Item 8, we had not yet come across controllers containing integral terms, and just looked at what proportional gain we needed.

However, if the plant contains an integral, then the only way in which the measured variable can stop moving is for the difference between the setpoint and measured value to return to zero. We therefore get a control which is accurate <u>without</u> using an integral term in the controller. We just make use of the integral in the plant instead.

You could possibly put an integral within your control program as well, but this gives little benefit and is rarely worth doing.

CASE 6: SYSTEM CONTAINING AN INTEGRAL AND A TIME CONSTANT

If you have a system containing both an integral and a time constant, then you can rely on the integral to reset the difference between the measured value and the setpoint to zero; you do not need an integral term in the controller. However, to get any reasonable sort of response out of the closed loop, you will certainly need a derivative term, and maybe a second derivative as well.

One instance of such a system is a flow control with a motor-driven valve with no valve-positioning servo. The analog output from the GEM 80 just goes to a power amplifier which feeds the motor. As long as there is any output from the GEM 80, then the motor will turn, and the valve will keep on moving. The valve position is therefore proportional to the integral of the GEM 80 output. On top of this integral term, we have the normal plant time constant and time delay.

To cater for this type of system, the PIDINC Special Function S35 is included in most GEM 80 controllers. Making use of the integral in the plant, it allows you to set up a 3-term, PID, control. The function itself, however, is really a P-D-D^2 control because it does not contain any integral term itself.

We previously remarked that derivative action tended to amplify noise (see Fig. 47). A second derivative (D^2) term is even worse. A further problem is that, for a motor-driven valve, there may be stiction. After a change in setpoint, you could finish up with a standing output from the PIDINC control which was insufficient to make the motor move.

To prevent these effects, GEM 80 controllers which include PIDINC also include the INCOUT Special Function S36. This only provides an output within the window between an upper and a lower limit, the window being repeated for both polarities. You set the lower limit to make sure that the minimum signal is sufficient to always make the motor move. You set the upper limit to correspond to top speed of the motor. However, INCOUT does not just limit the signal but gives an output whose integral is equal to the integral of the input from the PIDINC function. Thus, although any spikes produced by the derivative and D^2 terms are limited by the INCOUT, their areas are taken into account.

For further details of PIDINC and INCOUT, refer to the software data sheet for these Special Functions.

----oOo----

INTRODUCTION

Some types of control use an output device which, instead of accepting an analog signal, needs an ON/OFF signal. A typical case is an electrically heated furnace, where you control the temperature by switching the power on and off repetitively to obtain the correct mean power input level. Obviously, such a control is only satisfactory where:

- The material or the fluid being heated has a long time constant which will maintain the temperature virtually constant in spite of the power being switched on and off.

- The repetitive switching on and off of the power is sufficiently fast for the given time constant and the given accuracy.

Controls of this type are generally referred to as "bang-bang" controls.

DITHER

Bang-bang controls must be able to switch the output element on and off as a variable mark-space ratio signal. While the power is on, the measured value will rise, and while it is off it will fall, as in Fig. 53:

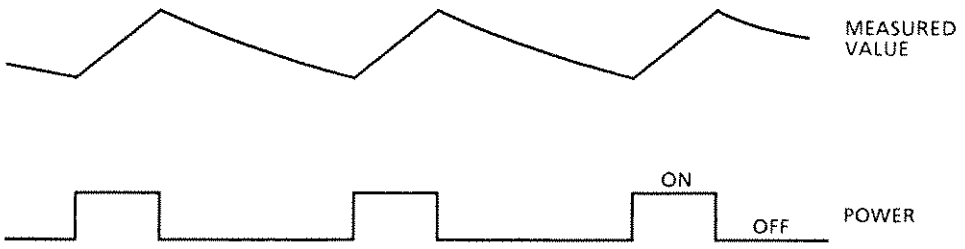

Fig. 53
Variation in measured value

The measured value will therefore dither up and down about the required value, rising and decaying along its time constant curve (see Fig. 22 and Fig. 23). However, if the time constant is long compared with the frequency of our output waveform, you can take the lines as virtually straight, and the waveform of the measured value as triangular.

Frequency

First, we need to assess at what frequency we need to switch the output for a given dither amplitude. If we make this frequency too low, then we will get an unacceptably large dither amplitude. With power off, the measured value will clearly tend to zero. With power on, it will tend to some maximum value. We can therefore look at a single triangle of the waveform as in Fig. 54.

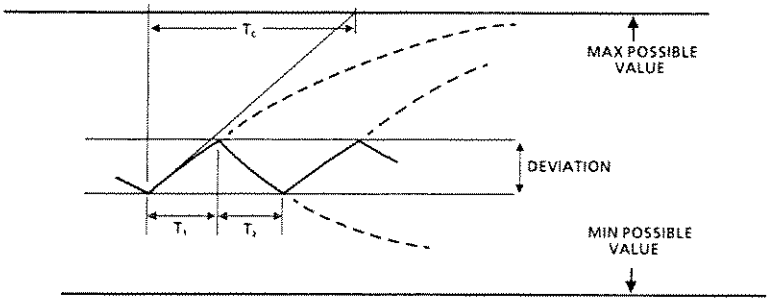

Fig. 54
Assessing the variation in measured value

If we don't want the measured value to vary by more than a certain amount (e.g. ±1%) then we can find what the minimum time T_1+T_2 must be. This in fact occurs when the measured value dithers about a value half way between zero and the maximum possible. (The maximum possible is the value which would be reached if the power were switched on continuously.) Thus, for ±1% variation, we need to find T_1+T_2 for the measured value moving between 49% and 51%.

As long as the deviation is small and about the half way value, then T_1 and T_2 will be equal and given by:

$$T_1+T_2 = 8 \times T_c \times \text{deviation}$$

where:
T_c = Plant time constant

Thus, for ±1% dither on a plant with a 120 second time constant, we would get:

$$T_1+T_2 = 8 \times 120 \times 0.01$$
$$= 9.6 \text{ seconds}$$

In this example, we can therefore switch the power on and off quicker than 9.6 seconds per cycle but, if we switch slower than this, then we will get more than ±1% deviation.

Depending on how fast you have to switch, you may need to use some form of static contactor for the output device. If you were thinking of using a conventional electromechanical contactor, then you should check whether the number of switching operations required will give a reasonable life. Otherwise, a static contactor is a necessity.

Resolution

Second, knowing what the cycle time is, we can find what resolution we will get. If, for instance, we had a 9.6 second cycle time as above, and the program repetition interval were 20 ms, then there would be 9.6/0.02 = 480 program executions during one cycle of the waveform.

For an output half way between zero and maximum output, we would get an equal mark-space ratio on the output, i.e. in our example, 240 executions ON and 240 executions OFF. The smallest increase in mean output we could therefore make would be to 241 ON and 239 OFF, or a change of 1 in 240, or just over 0.4%.

If your plant has a much longer time constant, then your cycle time might be a lot longer than the 9.6 seconds in our example. You might then get quite a good resolution using a repetition interval, for the rungs working out the mark-space ratio, of 0.1 second, 1 second, or even longer. You don't necessarily have to execute the particular rungs in your GEM 80 program on every pass through the program. Instead, you could put them in a BLOCK operated at regular time intervals.

HOW TO PRODUCE THE DITHER

To get dither of the frequency we have been describing on the previous page, we cannot rely on the closed loop system to do it for us. The response of the loop could be much too slow. Any time delays would cause the measured value to overshoot by some distance before any resulting change in output to the plant had any effect.

Instead, we must produce dither open loop. We certainly need a closed loop control, but we feed the output of any proportional, 2 or 3-term control to a "waveform generator" which produces the mark-space ratio ON/OFF signal. The closed loop control function therefore only has to alter the input to the waveform generator to eliminate any long term drift, but does not have to respond any quicker than on a control system with an analog output.

To produce the variable mark-space ratio output, you could make use of the DELAY function. However, this can only operate in increments of 0.1 second on most GEM 80 controllers. If this is too coarse for the amplitude of dither you can accept, then you can make a waveform generator as in Fig. 55 below.

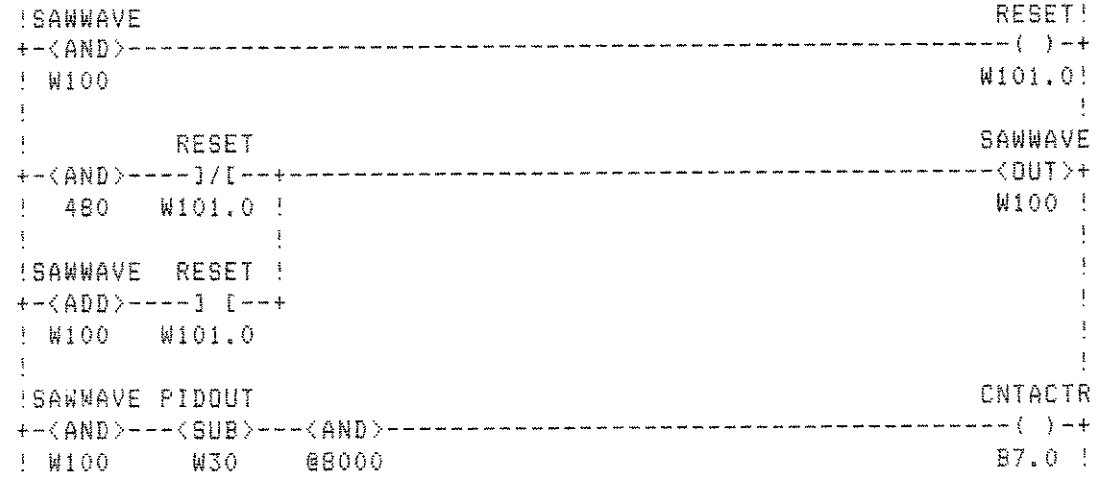

Fig. 55
Mark-space ratio signal using waveform generator

In this section of program, the first rung picks up the coil W101.0 whenever W100 is non-zero. Since on the first time through the program W100 will in fact be zero, then initially W101.0 is dropped out. Therefore, when the second rung is encountered, W100 is set to the 480 value.

On subsequent executions of the program, since W100 now contains a non-zero value, the coil W101.0 will be picked up. When the second rung is executed, the top branch of the rung will be ignored and the second branch used. Note particularly that it begins with an ⟨ADD⟩ (not an ⟨AND⟩), which means that -1 from the left hand goal post is added to W100, and the result put back into W100. Consequently, the contents of W100 count down by 1 on each program execution until they reach zero, after which W101.0 drops out again and W100 gets rest to 480.

Thus, the value held in W100, if you plotted it against time, would be as below:

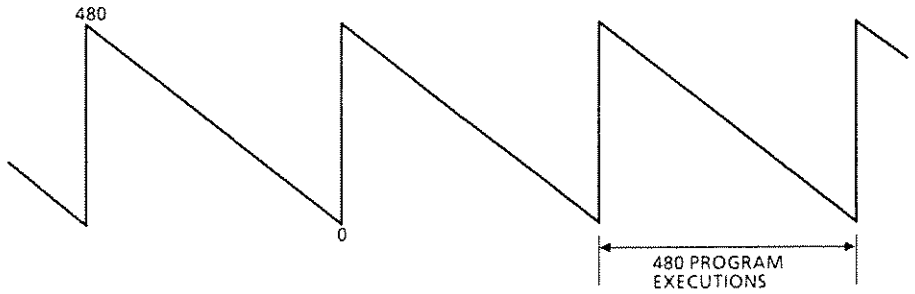

Fig. 56
Sawtooth waveform from waveform generator

The third rung compares the input from, for instance, a 3-term control function (mnemonic PIDOUT), against the waveform above. As long as the sawtooth is greater than the input, then the result of the ⟨SUB⟩ will be positive. However, once the sawtooth wave drops below PIDOUT, then the result of the ⟨SUB⟩ will be negative. We detect this by looking at bit 15 only, by masking with @8000. If this sign bit is set, then the output B7.0 is energized. The cycle then repeats when the next sawtooth of the waveform is encountered.

A different form of mark-space ratio generator is given in the software data sheet for the Special Functions PIDABS, PIDINC and INCOUT using the DELAY function. This has a lower time resolution than the example described above.

TIME DELAY INTRODUCED BY THE MARK-SPACE GENERATOR

If we consider a small change in input signal to the waveform generator, then we might be lucky - the output might be close to switching on, in which case we could alter its switch-on time almost straightaway. On the other hand, it might already have turned on, in which case we might have to wait almost a full cycle before we could alter the switch-on time.

Consequently, the average delay we can expect is half a cycle. This is the time we should use (in addition to any transport lags) for the plant time delay when calculating what proportional and integral gains we need in the control function.

Use of a derivative term on a bang-bang system of the type we have been describing is generally of no benefit. Any odd spikes produced by a derivative term may cause the output device to switch on and off for a single program execution. This is undesirable, especially if you are using an electromechanical contactor as the output device.

SLOW DITHER

As with conventional analog control loops, you can get unwanted dither occurring at a much lower frequency than that produced by the mark-space generator. It follows the same pattern as already discussed in Item 13, and you can use the same methods if you need to get rid of it.

Note, however, that the resolution of your mark-space generator can effect slow dither. If, for instance, your mark-space generator has a resolution of only 1%, then your control might want to provide an output of, let's say, 45.4%. Since your mark-space generator can only provide either 45% or 46%, then the best that the control can do is to cause the output to drift up and down between these two levels. If you needed less variation than this, then the best you could do would be to execute your mark-space generator rungs on every program cycle, and avoid putting them in a time-dependent BLOCK.

SUMMARY ON BANG-BANG CONTROLS

1. Work out the cycle time you need, using the formula on Page 80, for the amplitude of dither which is acceptable.

2. To get a given output amplitude resolution, work out what time resolution you need. This will tell you whether you need to execute your mark-space generator rungs on every program execution, or whether they can be put in a time-dependent BLOCK.

3. Use half the mark-space waveform cycle time as the time delay introduced by the mark-space generator when working out what proportional and integral gains you require in your control function.

-----oOo-----

SECTION 2

CONTROLS WITH MORE THAN ONE LOOP

CONTENTS

(continued)

----oOo----

CONTROLS WITH MORE THAN ONE LOOP

In Section 1, we looked at the design of open and closed loop controls containing a single loop. Most systems, however, contain more than one (open or closed) loop.

Once you have become familiar with the topics covered in Section 1, the design of a single loop system can become a fairly routine matter, where you just apply the rules without a great deal of thought. How you combine a number of separate controls into an overall system, however, does require some thought. It is the art of systems design, and provides the subject matter for this Section.

This type of design work tends to be much more application dependent. We can't tell you how to design a system for every possible application because there are just too many. Further, our customers are always finding new applications for GEM 80 Controllers, such is their flexibility.

This Section is therefore something of a potpourri of types of control with more than one loop. We include in this Section some examples of controls combining an open loop control with a closed loop control, i.e. we use the term "loop" somewhat loosely.

ON-LINE ADAPTATION

A most important feature of the pre-written software routines referred to as Special Functions is that their gain settings, limit values, etc., can be altered while the control system is running. This is a feature which we didn't exploit in Section 1, where we just worked out what settings you required for constant operating conditions.

For instance, with a simple analog 3-term controller, you have knobs or raise/lower pushbuttons to set up the proportional band, integral and derivative times. With the GEM 80 PIDABS Special Function S34, however, these settings are held as values in table locations. It is therefore a straightforward matter to adjust these values on-line to suit the particular operating conditions by putting suitable rungs in your control program.

In this Section, you will find many examples of the use of Special Functions. You should have been provided with a Software Data Sheet for each of the Special Functions contained in your particular model of GEM 80 Controller. You may find it helpful to refer to these sheets while you are reading this Section. However, although these individual data sheets give examples of how these functions can be used, you only begin to appreciate some of the features of these functions from seeing more examples, in more systems than can conveniently be described in individual data sheets. We hope, therefore, that this Section will give you plenty of ideas on the ways in which you can apply Special Functions, as well as ideas on system design.

----o0o----

INTRODUCTION

In Section 1, we assumed that the operating conditions for any of our closed loop systems were constant. In practice, they may vary.

For instance, if we have a flow control, we will probably find that the flow does not vary linearly with valve opening. We referred to this possibility in the very first Item on "Open Loop Control". To get round the problem, we used the function generator Special Function FGEN in Fig. 3 (Page 5). However, we have not so far considered what we should do in a closed loop control.

Operating conditions may also be influenced by some other variable outside our closed loop. This other variable may be controlled by some other independent closed loop system. For instance, if we are trying to control the temperature of a fluid at the outlet of a heat exchanger, then we will find that the plant time constant and the plant time delay may vary with flow rate.

We therefore have two cases to consider:

- Where the variation comes from non-linearities within our closed loop.
- Where the variation comes from another variable, probably controlled by another closed loop.

SLIGHT NON-LINEARITIES

If the time constant, time delay, or gain of the plant is going to vary by a only few percent, then we can treat this variation as just another form of disturbance which our closed loop system will look after. For instance, if the gain of the plant varies as shown below:

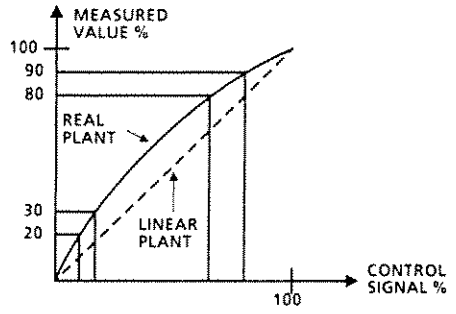

Fig. 57
Slight non-linearity in plant gain

then, if we wanted to change the plant output from 80% to 90%, our GEM 80 controller would have to give a slightly larger change in output signal than if we wanted to change the output from 20% to 30%, say.

If we used just a proportional control, then the non-linearity would be reduced M+1 times. If we drew a graph of plant output

against setpoint to our closed loop, we would then find that it deviated from the dashed line much less. If, as is likely, we included an integral term as well, then there would be no difference at all between the setpoint and the measured value, and the non-linearity would be completely eliminated.

LARGER NON-LINEARITIES

Even if the non-linearity in the plant characteristic is more marked, an integral term will still make the measured value and the setpoint equal, and thus eliminate the non-linearity. However, although the control is then O.K. for accuracy, its response may not be satisfactory under all conditions. For particular operating conditions, it could even be unstable.

In Section 1, we showed how to work out the proportional gain k_p to give us a loop gain M. Then, the integral gain k_i and the derivative gain k_d could both be worked out depending on k_p. The value we used for k_p was M times the open loop gain. We also noted that there was a maximum value we could set for M if we wanted a non-oscillatory response.

However, if our system does not have a fixed open loop gain but a gain that varies like Fig. 57, only more so, then how should we proceed ?

Note that, if we had a plot of the plant gain, we would be able to find out the minimum and maximum gains by drawing tangents, as in Fig. 58:

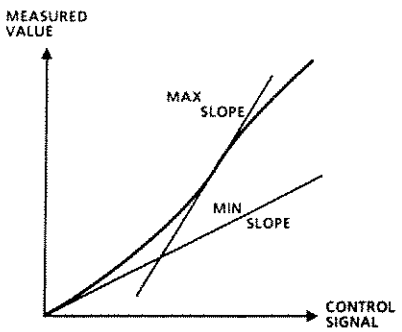

Fig. 58
Finding the maximum and minimum plant gains from a plot

Change of Gain Only

If nothing changes except the gain (i.e. time constants and time delays remain fixed), then we have two options:

- Set up the control so that it will be stable for both maximum and minimum gains. Using this method, the response will not necessarily be optimal in either case.

- Include a function generator (FGEN Special Function) to linearize the system sufficiently for the response to be virtually unchanged under different operating conditions.

For the first method (with no FGEN), we must set up the proportional term (and derivative term, if we have one) to cater for the highest gain condition. On the other hand, we must set up the integral term for the lowest gain condition.

We are likely to get responses something like those in Fig. 59.

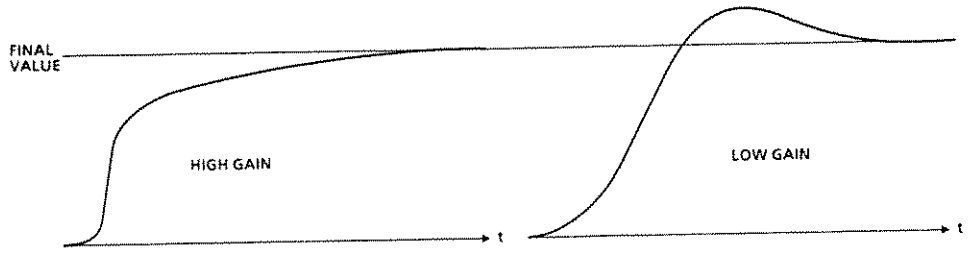

Fig. 59
Responses without any linearization

Note that, for the high gain condition, we get a fast initial response but a slow integral action. However, for the low gain condition, we get a slower initial response but with a better matching integral action.

For the second method, we put in an FGEN Special Function to keep the loop gain more or less constant. We can then obtain identical, optimal responses for all values of setpoint. However, we don't need to keep the gain quite as constant as we did for open loop control, since the action of the closed loop will still tend to linearize the system. We can therefore use an FGEN with only a few segments. The FGEN will be much easier to set up than for open loop control, as it will not need calibrating to the same degree of accuracy.

We need to place the FGEN after the control function, as in the rung below:

```
! SETPT     M/V    PIDABS            FGEN                          ACTR !
+-(AND)---(SUB)---SPEC.---VALUE---SPEC.---VALUE--------------(OUT)+
!  W10      W20      S34     W100     S30     W120                 B7  !
```

Fig. 60
Placement of linearizing function generator

By putting the FGEN in this position, the combination of the FGEN and the plant should have approximately a linear, constant gain characteristic, so that the PIDABS is operating as though it were driving a linear plant.

(Note that you shouldn't set up the limit settings of the PIDABS until after you have set up your FGEN to its proper shape, otherwise these limit settings could be wrong.)

To work out the shape of the FGEN, first you need a plot of the plant characteristic, plotted as measured value W20 against the GEM 80 output B7 to the actuator. You could do this by running, without any FGEN, either open or closed loop. If you run closed loop, then you may need to use low gain settings to prevent instability.

We are primarily interested in how much the gain varies (i.e. by how much the slope of the plot varies), not the precise shape of the curve. We can then determine what changes of gain to put into the FGEN to keep the overall gain more or less constant. To get an adequate plot, you may only need to plot 3 or 4 points (e.g. at minimum and maximum measured value, and 1 or 2 points midway in the span), after which you may well be able to draw in the rest of the curve to sufficient accuracy using a flexicurve.

To find the shape you need for the FGEN, it is easiest if you plot the plant characteristic with the axes arranged as in Fig. 61:

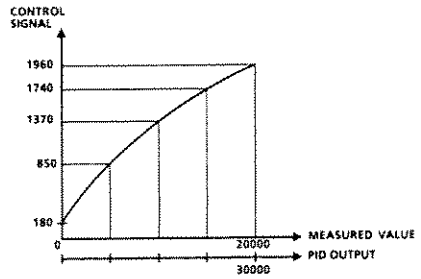

Fig. 61
Plot of the plant and FGEN gain characteristics

This Fig. shows the shape you need for the FGEN, but not necessarily its horizontal scale. We have plotted Fig. 61 so that the horizontal axis covers the span for the measured value W20 which, in this instance, is 0 and 20000. Now suppose that, for closed loop control, you want the PIDABS to give an output to the FGEN of 0 to 30000. We therefore take the plot and redraw the scale as 0 to 30000, covering the same range that was previously covered by the 0 to 20000 measured value range.

We now need to subdivide this 0 to 30000 range into a suitable number of equal increments. For each of the straight line segments produced by our FGEN, we need to check that the variation in slope of the actual curve does not differ too much from the slope of the segment. In most cases, up to +30% difference will not affect the response or stability of the system to any noticeable extent.

In Fig. 61, we have divided the range into four segments each 7500 wide. We then measure off the B7 values at the break points, and this gives us our FGEN characteristic. The required table values in this example will be as follows:

```
W121 :     0      (= leftmost point)
W122 :  7500      (= size of increment)
W123 :     5      (= quantity of points = segments + 1)
W124 :   180      (= 1st ordinate)
W125 :   850      (= 2nd ordinate)
W126 :  1370      (= 3rd ordinate)
W127 :  1740      (= 4th ordinate)
W128 :  1960      (= 5th ordinate)
```

Now, the FGEN values given above, and the corresponding segments in Fig. 61, may look a fair approximation to the curve. The deviation at the centre of any segment is less than 20, or about 1% of full span. For response, however, we are interested in how well the segments follow the slope of the curve, and not just what the actual deviation from the curve is.

In fact, the rightmost segment is not very good. At its left hand end, the curve is over 30% steeper than the segment and, at its right hand end, the curve is over 50% flatter. We might find that, to make the segments follow the slope of the curve better, we needed to use a smaller size of increment (perhaps W122 = 5000, giving us six segments rather than four). This would keep the gain, and therefore the response, more nearly constant under all conditions.

Change of Gain and Time Constant Together

In some types of process, although the gain of the plant is non-linear, the time constant changes proportionally with the gain.

If the plant time delay remains fixed, then you can generally design your system for such a process without any linearization, and you don't then need an FGEN Special Function. Remember that the loop gain you can use depends on the ratio of the plant time constant to the plant time delay (see Page 29). As the time constant increases, you can therefore also increase the loop gain. If you have a process where, as the plant time constant increases, the plant gain also increases, then the loop gain is going up naturally, without you having to do anything about it in your GEM 80 program.

A typical case is on the flux control for an electric coil (e.g. for a motor field control) where, as the iron is driven towards magnetic saturation, you need a larger change in voltage to achieve the same change in magnetic flux (i.e. the gain has gone down). However, you also find that the time constant has dropped proportionally to the gain. You therefore do not need to include any linearization in your flux loop, and you will get virtually the same response shape whether the flux is at low or high values. See the Fig. 62.

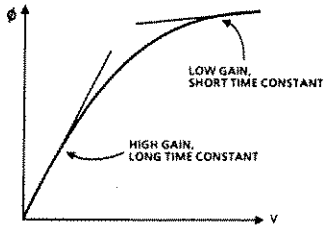

Fig. 62
Flux/voltage characteristic of an iron-cored coil.

CHANGE OF PARAMETERS WITH AN EXTERNAL VARIABLE

As an example, we will consider a temperature control where the flow through the heater or heat exchanger varies. Exactly similar types of control can arise, however, for numerous flow processes, such as a tension control on a rolling mill with varying strip speed.

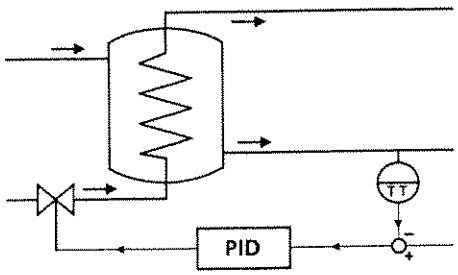

Fig. 63
Heat exchanger temperature control

Time Delays

In any flow process containing a transport lag, this lag will vary inversely with the flow. That is, the faster the flow, the shorter the time delay.

For our example, we may well have a transport lag because of the temperature transducer being some distance downstream of the heater or heat exchanger. To compensate for the change in transport lag, we may be able to use a flow measurement signal to modify the gains in our temperature control. (As we probably already have a closed loop flow control containing a flow transducer, we may have such a signal available without needing to add an extra transducer.)

However, if a major portion of our time delay comes, not from a transport lag, but from the program repetition and sampling intervals, or the sum of several small filter time constants (see Pages 32-34), then any change in flow will not affect the time delay to the same extent.

Time Constants

In our example, changing the flow will also alter the time constant. We referred to this effect earlier in Item 5 (Page 36). If we increase the output flow rate from a heat exchanger without altering the heat input, we have effectively increased the losses (i.e. the heat is being taken away quicker), and so the time constant becomes shorter.

Gain

Again in our example, altering the flow will change the plant gain as well as the time delay and the time constant. When the flow is greater, we will obviously get a lower outlet temperature, in the steady state, for the same heat input.

CONTROLLER WITH A FIXED SETTING

Since, in our example, the gain and time constant are varying together, we have at first sight a system like that described on Page 92. With no change to the controller settings, we might expect to get very similar responses regardless of whether the flow was low or high.

However, what is different in this case is that the time delay is also varying. With a fixed controller setting, we must therefore set up the controller to give a reasonably stable response for the maximum time delay (see Page 29).

Here we have a problem. When there is no flow, the time delay becomes infinite. We must therefore arrange in our GEM 80 program to lock out the temperature loop until the flow reaches some minimum operating value. At this minimum operating flow, we will then obtain the maximum operating time delay.

With a fixed controller setting, we must set up the control for this minimum operating flow condition. We will then get similar responses at higher flows, although the time delay will become shorter as the flow increases.

CONTROLLER WITH ADAPTATION FOR FLOW

To get a faster response when the flow is high (when the time constant and time delay are both short), we need to modify the gains of our PI or PID control function.

If all the time delay came from a transport lag, then we could make the proportional gain vary directly with flow. However, there will in practice always be some fixed portion of time delay to add on to this, i.e. there will always be sampling and controller delays (see Page 33). Further, as we noted above, the transport delay tends to infinity as the flow tends to zero.

If we know how far downstream the transducer is, and what the flow velocity is (= volume flow rate/pipe cross sectional area), then we can work out the transport lag as:

$$T_L = \frac{D}{V}$$

$$= \frac{D\ A}{R}$$

where:

D = distance downstream
V = velocity
R = volume flow rate
A = pipework cross sectional area

In addition to the transport lag, we will have a fixed delay from our controller program and from sampling (see Page 33). The total time delay is therefore:

$$T_d = \frac{D\ A}{R} + T_k$$

where T_k is the delay from the controller + sampling + filters.

For a proportional control, we need to set the proportional gain to vary inversely with T_d. In other words, we need a gain k_p of:

$$k_p = \frac{1}{\frac{k_1}{R} + k_2}$$

(All we have done here is to take the reciprocal of T_d and multiply by a constant).

Now, if the flow rate R becomes zero, then k_1/R becomes infinite, so k_p becomes zero. If we can therefore implement this formula in a GEM 80 program, we will automatically lock out the proportional control term when the flow is zero. We can implement it either with a couple of divisions (using LINCON for accuracy rather than DIV), or else with a function generator FGEN Special Function.

However, we also need to modify the integral term. From the formulae on Page 56, you will see that the maximum value for k_i depends on k_p and on $1/T_d$. Since k_p already depends on $1/T_d$, this means that k_i is proportional to $1/T_d^2$, or to k_p^2. Just as for the proportional term, the integral gain will become zero when the flow is zero.

If we have any derivative term, however, it works the other way. Since its gain should be proportional to k_p and to T_d (not $1/T_d$), then the gain of the derivative term can remain fixed, and we do not need to alter it with changes in flow.

Whether we use multiplications or divisions, or set up the functions using FGEN Special Functions, the shapes of the functions we need for modifying the proportional and integral gains are as given in Fig. 64.

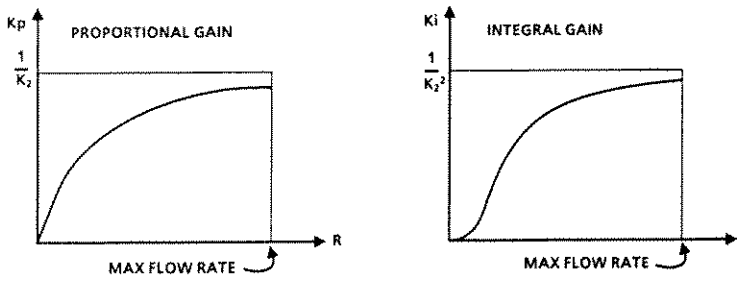

Fig. 64
Functions for adapting P and I terms with flow

If you set up these functions with FGEN Special Functions, then you could implement adaptation with flow using a program as below:

```
! FLOW    FGEN                                                  PROP !
+-<AND>---SPEC.---VALUE-----------------------------------------<OUT>+
!  W30     S30     W350                                         W301 !
!                                                                    !
! FLOW    FGEN                                                   INT !
+-<AND>---SPEC.---VALUE-----------------------------------------<OUT>+
!  W30     S30     W400                                         W302 !
!                                                                    !
! SETPT    M/V   PIDABS                                         ACTR !
+-<AND>---<SUB>---SPEC.---VALUE---------------------------------<OUT>+
!  W10     W20     S34     W300                                  B7  !
```

Fig. 65
Program with adaptation of P and I terms with flow

Transient changes in flow

If we suddenly change the flow, then we would like the temperature loop not to see any disturbances, but to maintain the measured temperature value exactly as it was before the flow change. To do this, the temperature loop will need to alter the heat output, without the measured value and setpoint changing.

Now, when the conditions in the temperature loop are steady, with no error between the setpoint and measured value, then all the output from the GEM 80 comes from the integral term. We could therefore modify the integral term. Since the heat required will, for a given temperature, be proportional to flow, then we could multiply the integral term by a number proportional to flow.

However, for stability, we found that we had to multiply the integral term by a value which was not directly proportional to flow. We therefore seem to have two conflicting requirements for

modifying the integral term. How, then, can we cater for the stability requirement and the transient flow change requirement at the same time ?

The answer is to perform two separate multiplications to the integral term, one before the counter forming the integral, and one after the counter. In concept, we want a system for the integral term with a block diagram as below:

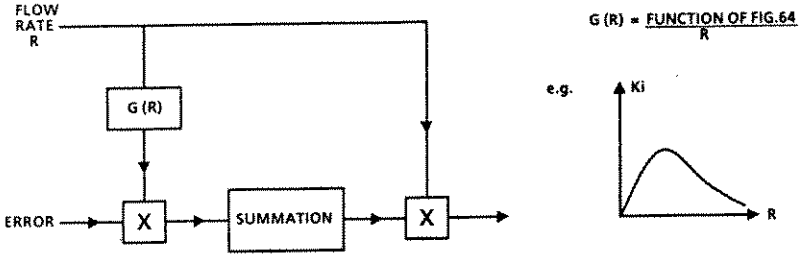

Fig. 66
Block diagram for modifying the integral term

If we consider the system in a steady state condition, then we have zero error as input to the integrator. Therefore, changing the function of flow G(R) makes no difference; you are just multiplying zero by a different value, and the input to the integrator is still zero. On the other hand, although the input to the integrator is zero, its output isn't; it will contain a value formed by counting up previous non-zero errors. Consequently, altering the flow input R to the second multiplier does make a difference to the final output. So, for steady state conditions, altering the flow causes the output from the multiplier after the integrator to alter directly proportionally to flow.

If we now consider the system with steady flow but the error varying, then changing the error signal causes a change in output which depends on both multiplication factors. We therefore get the correct integral gain for stability of the temperature loop.

However, most 3-term control algorithms use only a single value for the integral gain, and not two values as we need to implement Fig. 66. With the S34 Special Function PIDABS, the multiplier comes before the counter. Further, we can't use a separate multiplier function such as LINCON after the PIDABS. If we did, this would not only change the gain but also affect the limit settings.

To get round the problem, we can take the previous output from the PIDABS function, multiply it by the flow rate and divide it by the previous flow rate. We must do this before the PIDABS is executed. A possible implementation is given in Fig. 67 on the next page.

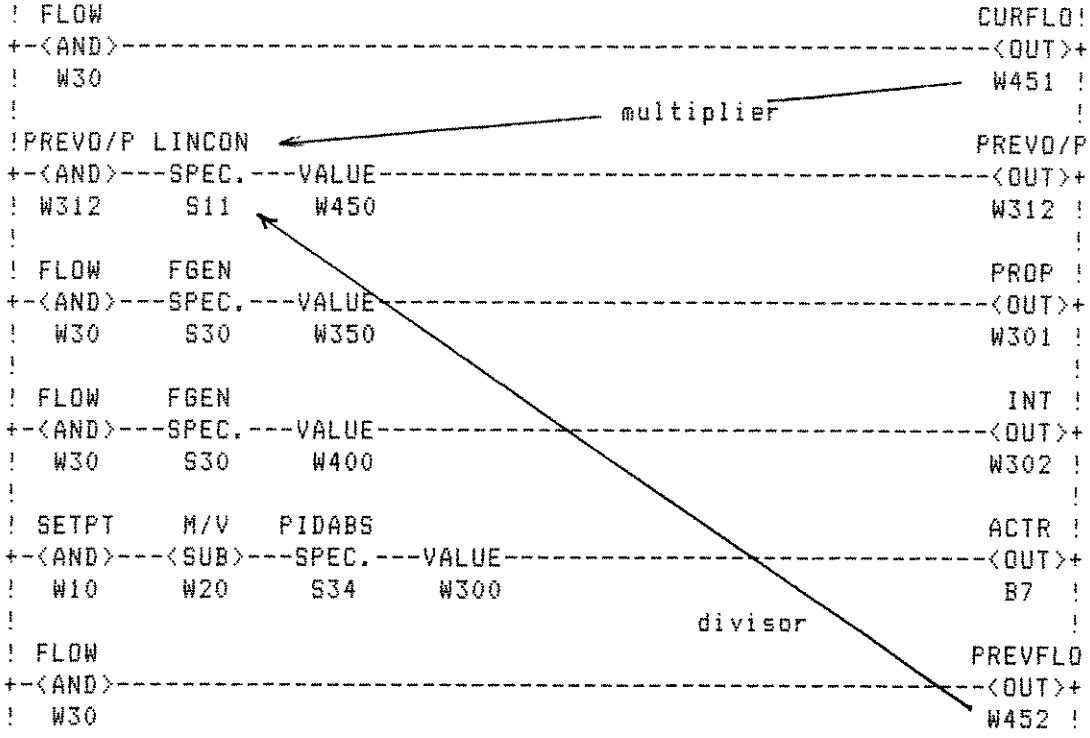

Fig. 67
Program with adaptation of P and I terms with flow

In this section of program, rungs 3 to 5 are exactly the same as those in Fig. 65, and the two FGEN Special Functions also have exactly the same table values as previously. Rung 2 is the one which looks after changes in flow with its LINCON Special Function. The previous output PREVO/P from PIDABS, from table location Zm+12 (i.e. W312 in this example), is scaled by the LINCON and put back into W312. The multiplier W451 for the LINCON is taken from the current value of flow in the first rung, and the divider W452 is taken from the last rung which, because it isn't executed till after the PIDABS function, will hold the value of flow as it was when the program was last executed.

Note that, if the flow rate is not changing, then W451 and W452 will contain equal values. The LINCON will then just multiply W312 by unity, and therefore the program will operate just like Fig. 65 did.

There are two remaining problems which come to mind. First, if we start off the system with zero flow, then the LINCON will attempt to multiply by zero divided by zero, which is mathematically indeterminate. However, if you try making the divisor zero with the LINCON Special Function, it assumes that there has been a programming error, and sets the output to zero, which is O.K. in this application.

Second, if we are modifying the value held in Zm+12 with flow, then we can, if the flow changes very rapidly, affect the rate limits. These decide by how much the output from PIDABS can increase or decrease on any execution, compared with the output on the previous execution. However, if we start modifying the value stored for the previous output, then the rate limits will not operate correctly. This will only present a problem in practice if the flow rate is going to change fairly rapidly. You might then need to put in an additional RAMP function after the PIDABS.

SELF TUNING CONTROLS

Self-tuning PID controls are generally of one of two types:

- Systems which tune off-line.

- Systems which tune on-line.

Systems of the first type generally operate in a different mode when tuning, i.e. they operate either in "Run" mode or in "Tune" mode. They therefore cannot control the plant properly while tuning. If you used a self-tuning algorithm of this type for the example we looked at above of changing the flow through a heat exchanger, then the operator would have to switch to "Tune" mode every time the flow was changed, to retune the PID control.

Systems of the second type tune on-line. To do this, they need some disturbance to the plant so that they can observe the response, and then alter the settings of the PID control to better values. If you tried using an algorithm of this type for our example then, each time you altered the flow, the algorithm would have to readjust the PID settings. However, it can only do this after there has been a disturbance where it can observe the response.

Consequently, although self-tuning algorithms for PID controls have their uses where the operating conditions are varying, you get better responses if you can directly modify the PID control settings, rather than waiting for a self-tuning algorithm to find them out for you. As we have seen above, it may take a little work to analyse the system, and to find out how the settings should be changed, but the end result is a quicker response system than you can get from self-tuning controls.

SUMMARY ON HANDLING PARAMETER VARIATIONS

1. The action of the closed loop compensates for slight non-linearities. You don't then need to take any further action.

2. If you have a larger non-linearity where the gain changes but the time constant and time delay remain constant, you can achieve a stable control if you set the proportional and derivative terms for the highest gain condition, and the integral term for the lowest gain condition.

3. To get responses which are nearer to the optimal, you can use a function generator to compensate for the non-linearity of the plant. You should arrange that the combination of the function generator and the plant has a virtually constant gain.

4. If you have a plant where the time constant varies proportionally to the gain, but the time delay remains constant, then you should get virtually the same response for different sizes of setpoint. You should not therefore need to put in any function generator.

5. Where plant characteristics vary with some other process variable, then you can modify the terms in a PID control as a function of this other variable. You should then be able to get the optimal response for any condition.

6. To minimize the disturbances which can be caused by the other process variable changing, you may need to change not only the gain of the integral term but also the stored value held by the integrator.

----o0o----

INTRODUCTION

By a cascade control, we mean a control where there is one closed loop control entirely enclosed by a second closed loop, as in the Fig. below:

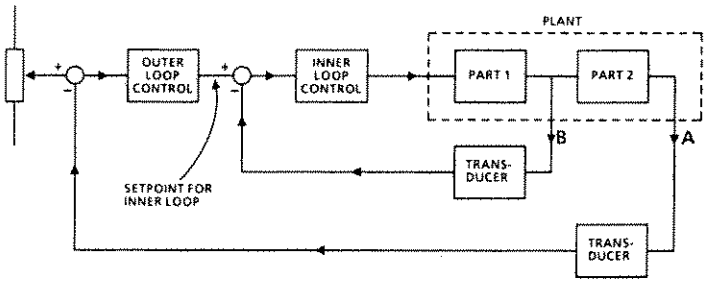

Fig. 68
Cascaded control loops

The outer loop measures the error between the setpoint and the measured value A, and from this works out (maybe using a PID control) what setpoint to provide for the inner loop. This is compared with the measured value B, and the inner control loop (maybe again using a PID control) works out the final signal to the plant.

As the overall system has only one setpoint (the one being compared with the measured value A) which is externally adjusted, we obviously need good reasons for going to the added complication of having an inner loop. Compared with a single loop system, we have to provide an extra measurement transducer for the measured value B, and an additional analog input channel.

There are a number of possible reasons for wanting a cascade control:

1. You might have had to place a measurement transducer in a location other than where you really wanted it, in order to get a fast response (see Item 5 on "Proportional Control Stability" under the heading "Transport Lags", Page 32).

2. During the operation of the control, you might want to make sure that some variable, other than the one you were controlling, did not exceed some limiting value (e.g. while you are primarily interested in controlling the speed of a motor, you want to also make sure that the current, torque or power does not exceed a certain value).

3. The plant might be very difficult to get a good response out of with a closed loop control (see Item 8 on "Special Cases" in Section 1, Page 46f).

These reasons are not necessarily mutually exclusive. You might find that you had to satisfy reasons (2) and (3) both at the same time, in which case a single inner control loop could probably be made to satisfy both requirements together.

SYSTEM WITH TWO MEASUREMENT TRANSDUCERS

Suppose we have a temperature control with two transducers, one a very short distance downstream of the heater, and one at the outlet from the pipework system. We mentioned (see Page 32) that we needed a transducer a short way downstream of where we inputted the heat so as not to get a long transport lag. However, the final outlet is where we really want to control the temperature. We therefore use a second temperature transducer to provide the measured value for the outer loop. As a block diagram, our system is as the Fig. below:

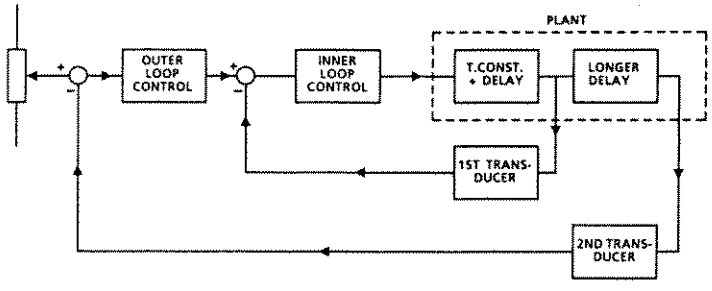

Fig. 69
System with inner and outer temperature transducers

The question is how to stabilize the outer loop. Now, having designed our inner loop, we know that its response will roughly be equal to a time constant and a time delay. If we have stabilized the loop as given in Section 1, then the time constant and time delay should be nearly equal. We can therefore redraw our block diagram, replacing the inner loop by a single block as below:

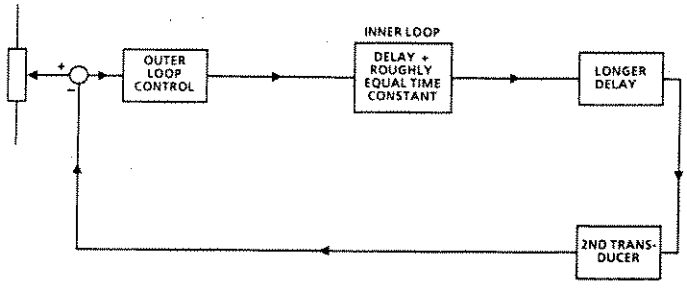

Fig. 70
Inner loop replaced by a single block

The remainder of the plant gives a longer transport lag, in the pipework to the second temperature transducer. We therefore get a system of the type we called our Special Case 2 (see Item 8, Page 45), where we have a transport lag (equal to the sum of the inner loop and pipework transport lags), plus a comparatively short time constant.

To make the system stable, we need either a long time constant or an integral term. The response we will get from this outer loop will then be a transport lag plus a time constant of roughly equal size:-

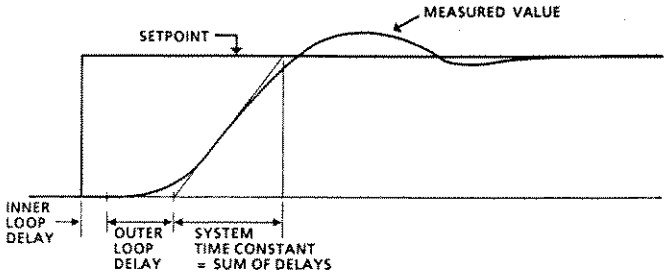

Fig. 71
Response of outer loop

This is the best we can do with straight closed loop control. It means that the response of the overall system to a change in temperature setpoint is rather slow. The only way to improve this is to use some feedforward to the inner loop. Feedforward is covered in the next Item.

You might have noticed that, if we hadn't used an inner loop, we would have been able to get just as fast a response to a change in temperature setpoint. However, the inner loop looks after disturbances much quicker. If there is a change in flow rate, for instance, the inner loop can adjust the heat input much quicker than a single loop system would.

INNER LOOPS FOR LIMITING PROCESS VARIABLES

To limit a process variable, there are basically two methods:

- Closed loop which only comes into play when the variable exceeds a set level (often referred to as a "spillover" system).

- Closed loop operational all the time, as the inner loop in a cascade control.

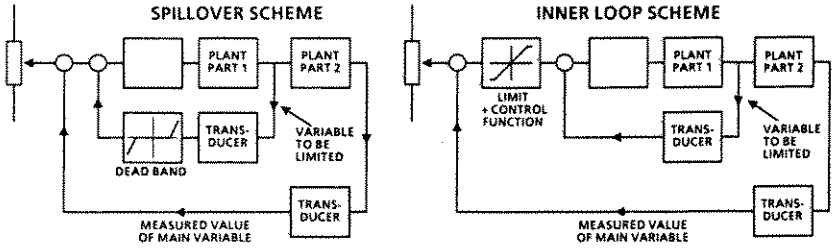

Fig. 72
Two methods of limiting a process variable

Although a spillover system may seem the more obvious way for the design of a system to limit a process variable, it presents problems. First, the variable must exceed the set level before any limiting action takes place, and therefore it is bound to overshoot this level. To overcome this, you could deliberately set the level down to a value below the limit value, but you then get a rather 'soft' limit, with the limiting action starting to occur early. Second, to keep the size of overshoot small, you want the variable to exceed the set value by only a small amount. You therefore need a very high gain loop, which may not be possible.

On the other hand, an inner loop can stop the process variable virtually dead, and can provide you with a hard limit with no overshoot. Further, you don't need a loop with as high a gain to achieve this as you would need for a spillover system.

If the outer loop can provide step changes of setpoint to the inner loop, you need the response of the inner loop to be fairly dead beat. You should therefore set the proportional and integral gains on the inner loop control to slightly lower values than given by the formulae given in Section 1. If the changes in setpoint coming from the outer loop are ramped, then the gain settings are less critical. If your outer loop uses a PIDABS in its control, then you can make the setpoint ramped by using the rate limits, rather than by using a separate RAMP function.

As long as the measured value of your inner loop follows the setpoint provided by the outer loop fairly faithfully, then you can also restrict the rate-of-change of the measured value. You can do this by ramping the setpoint with rate limits or a RAMP function. Limiting rates-of-change is generally difficult to achieve with a spillover system.

Next, consider how we stabilize the outer loop. As for the case we looked at earlier of a system with two transducers measuring the same quantity, we can reduce the inner loop to a single block. This will have a response consisting of a time delay and a time constant of roughly equal size. We can now add this to the response of the rest of the plant, just as we did for the previous example in Fig. 70. In this case, however, it is likely that the remainder of the plant has some time constant as well. We therefore get a system as in Fig. 73.

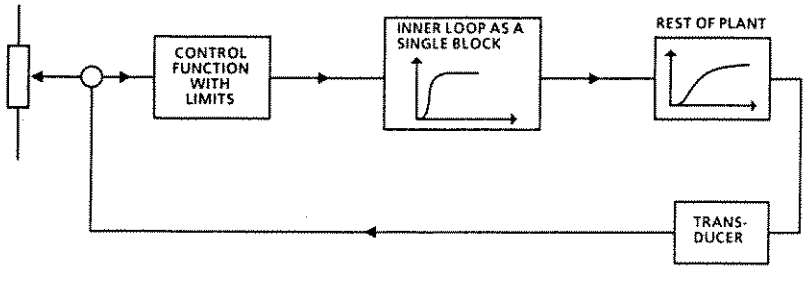

Fig. 73
Equivalent system with Inner Loop as a Single Block

We can now combine the rest of the plant with the inner loop. It is most important to note that:

- You can add time delays together.

- You can't add time constants together.

Instead, to a first approximation, you must add the smaller of the two time constants to the time delay. We therefore obtain an overall response consisting of:

- A time constant
  = larger of the two time constants. It might be either that of the rest of the plant, or that of the inner loop.

- A time delay
  = time delay of inner loop + time delay of the rest of the plant + the smaller of the two time constants.

Example

Suppose that the first part of the plant has a 10 second time constant and a 1 second time delay, while the second part of the plant has a 20 second time constant and a 2 second time delay.

By putting an inner closed loop control round the first part of the plant, we should be able to get a response from this loop where its effective time constant is reduced to a value equal to the time delay, i.e. about 1 second. We therefore have the following values to combine:

Inner loop:	1 second time constant
	1 second time delay
Rest of plant:	20 second time constant
	2 second time delay

By the rules given above, we get a combined response of:

20 second time constant	(The larger of the 1 second and 20 seconds time constants)
4 second time delay	(= 1 sec. time delay + 2 sec. time delay + 1 sec. time constant)

This, to a first approximation, is the combined response of the inner loop + the rest of the plant. It is the response of the effective plant round which we then put our outer closed loop control. We can therefore use these values when working out our PID gains for the outer loop.

Since we should be able to get a response where the time constant is roughly equal to the time delay, the effective response to changes in setpoint for the outer loop should roughly be a 4 second time constant with a 4 second time delay.

SYSTEMS WHICH ARE DIFFICULT TO STABILIZE

The example we have just worked through is interesting from the point of view of what response we might have got if we hadn't used an inner loop. Our plant would then have consisted of:

First part of plant:	10 second time constant
	1 second time delay
Rest of plant:	20 second time constant
	2 second time delay

If we combine these together, we get:

20 second time constant	(The larger of the 10 second and 20 second time constants)
13 second time delay	(= 1 sec. time delay + 2 sec. time delay + 10 sec. time constant)

Because the time constant is only slightly longer than the time delay, then if we put a single loop round this plant, we would only be able to get a very low loop gain, so that transient disturbances would hardly be attenuated at all. Further, the response we would be able to get to changes in setpoint would consists of a 13 second time constant + a 13 second time delay, to a first approximation. This is over three times slower than the response we were able to get with an inner loop present.

You can therefore obtain much improved responses on a plant which has two time constants of nearly equal size if you use an inner loop around one of them. We referred to this in Item 8 on Page 46. If, on the other hand, the plant has two time constants which are very much different in size, then you won't gain much improvement in response by using an inner loop.

However, regardless of whether it will improve the response or not, you may still need an inner loop to provide limiting of a process variable, as described previously.

SUMMARY ON CASCADE CONTROLS

1. You may need to use an inner loop, in spite of the additional cost of a transducer and analog input channel for it, to provide limiting of an internal process variable, and/or to improve the response of the system.

2. When you have designed your inner loop, you can reduce it to a single time constant and a single time delay, and then make use of these values when you design the outer loop.

3. To combine these inner loop values with the time constant and time delay of the rest of the plant, you must treat the smaller of the two time constants as a time delay, adding this to the other time delays.

----o0o----

INTRODUCTION

In some types of control system, you may want the plant to respond rapidly to a changing setpoint. However, you may find that, just using a conventional closed loop control, you can't get as fast a response as you would like.

To get round the problem, you may therefore need to add to the setpoint an anticipatory signal which gives the system a kick on a setpoint change. Because you are doing this outside the closed loop, then you will have to set up the amount of anticipation open loop. If you have too little anticipation, then the plant output will lag behind the setpoint. If, on the other hand, you have too much anticipation, then the output may overshoot.

RAMPED SETPOINTS

If you have a system where the setpoint comes from a RAMP Special Function S33, then this function also provides a rate-of-change output. You can therefore add on an appropriate amount of this signal to provide anticipation. Fig. 74 shows the way a plant might respond to a ramped setpoint, without any rate-of-change signal at (a), with the correct amount of rate-of-change signal at (b), and with too much at (c).

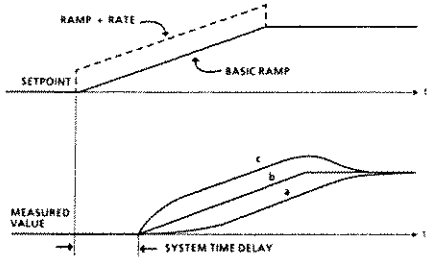

Fig. 74
Plant response to a ramped setpoint with anticipation

To implement this in a GEM 80 program is quite straightforward, as below:

```
! SETPT   RAMP                                                    RMPDS/P
+-<AND>---SPEC.---VALUE---<ADD>-------------------------------<OUT>+
!  W10     S33     W100    <   >                                 W15 !
!                        +-<   >                                     !
! RATE   LINCON          !                                         !
+-<AND>---SPEC.---VALUE-+                                         !
! W112     S11     W120                                            !
```

Fig. 75
Rung for ramped setpoint with anticipation

Here, the 12th table location used by the RAMP (W112 in this example) provides the rate-of-change signal (see RAMP Software Data Sheet). After scaling this signal as appropriate with a LINCON, you can then add it to the output of the RAMP.

There are a few points worth noting about the responses shown in Fig. 74:

1. We can't get any response from the measured value till after the effective system time delay, no matter how much anticipation we use.

2. If we exactly compensate just for the effective system time constant as in (b), then the shape of the measured value is a ramp of the same shape as the setpoint ramp, but lagging behind it by the system time delay.

3. By adding more anticipation as in (c), we can get the measured value to catch up with the setpoint ramp. However, we then get overshoot at the end of the ramp. Also, while the measured value is initially catching up with the setpoint ramp, it is changing at a faster rate than the ramp. In many types of process, neither of these two effects may be desirable.

The final point to note is that an anticipatory feedforward signal of the type we have been considering is an open loop compensation signal. We can set it up exactly for one operating condition of the plant but, if the plant parameters subsequently change, the degree of anticipation we have set may not then be correct. If we want to use fairly accurate anticipation, then we might have to:

- Make sure that the plant response was constant. To do this, we might have to linearize the plant, as given in Item 16, Pages 90-92.

- Modify the degree of anticipation with an external signal. For instance, with the LINCON in Fig. 75, we could use process variables (or functions of them) for the multiplier or divisor, rather than using preset constant values.

DERIVATIVE TERMS ON SETPOINT INPUTS

In theory, we could get the measured value to follow the setpoint with only a time delay and with no time constant, regardless of whether the setpoint was from a RAMP Special Function or not. To do this, we need to add a proportion of rate-of-change (derivative) of setpoint onto the setpoint itself.

Where we have no Special Function like the RAMP used in Fig. 75 which will directly provide us with a rate-of-change signal, we must derive it ourselves using separate instructions. We already saw how to do this in Fig. 45, Page 65. Fig. 76 shows how we can take the setpoint, and add on a proportion of rate-of-change to form the modified setpoint (MODS/P). The second rung is used to remember the last value of setpoint (LASTS/P) so that the change in setpoint since the last program execution can be found, and hence the rate-of-change.

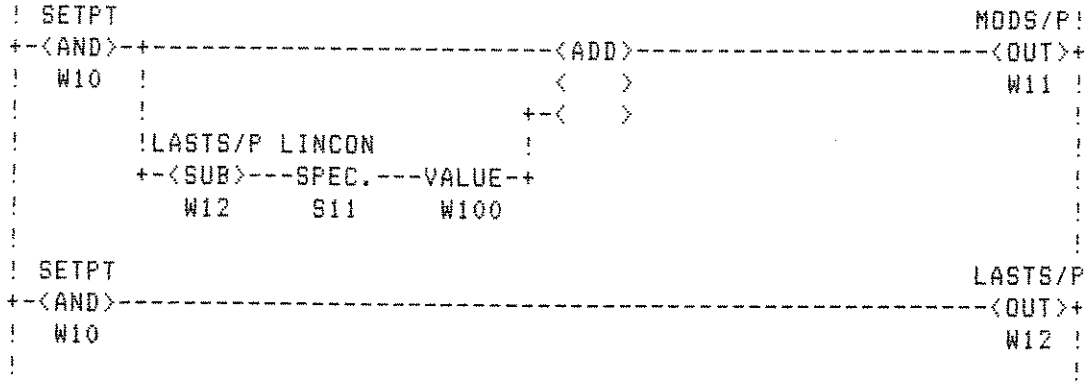

Fig. 76
Addition of rate-of-change to setpoint

In practice, however, we might not get the improvement we expected. For instance, the measured value would fail to follow a step change of setpoint as a step. The response will in fact be limited by the actuator or other controlling element on the plant.

For instance, if we have a temperature control, there will be a maximum heat input we can provide to the boiler or heat exchanger. This could be determined by the maximum steam flow in a heating pipe, or by the maximum power into an electric heating element. Once we have reached this maximum, there is nothing further our control system can do. The maximum heat input, and not the control system, determines the fastest rate at which the temperature measured value can change.

We obviously cannot get a step change in temperature, even though we may be able to get it to change quite fast. If we want faster responses, then we must modify the design of the plant, by putting in a heater which can transiently provide more heat when we want a fast temperature change. The heater won't need to provide this amount of heat during steady running conditions.

Where we use a controlling element which can transiently provide excess power, we often talk about the "forcing" it can provide. For instance, if a system can transiently provide 4 times the amount of power required for steady state running, then we say that it has 4 times forcing. Fig. 77 below shows the effect of forcing on the rate of change of measured value.

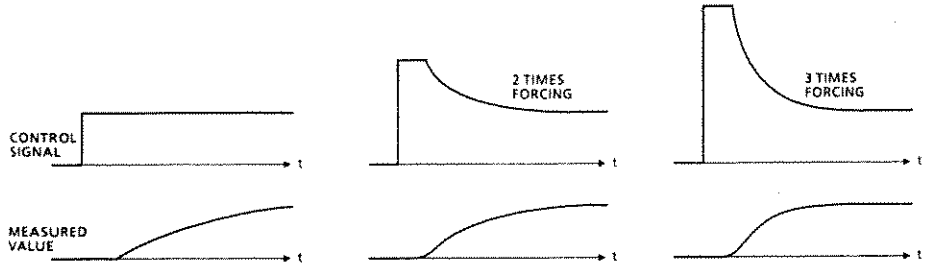

Fig. 77
Effect of forcing on response to large setpoint changes

Suppose that we add a rate-of-change onto a setpoint to speed up the response of the measured value. If we try giving the system large step changes of setpoint, the actual response we get will depend, not only on the setting of the derivative term, but on the degree of forcing as well.

Note that the amount of forcing we have available mainly affects the response on large changes of setpoint. For small changes of setpoint, we can often get an adequate response even with only unity forcing.

To speed up the response on a drop in setpoint, we need downward forcing. This is more difficult to provide than upward forcing. For instance, on a temperature control, if you want to push down the temperature fast, the best you can usually do is to cut off all heat input. If you want to reduce the temperature any faster, then you will need some form of cooling, i.e. some form of heat removal. It is not often economic to provide such a separate cooling system unless response requirements are very critical.

On the other hand, with electrical control systems such as current controls, you can often find types of power convertor which can regenerate power into the supply. You can then rapidly force the current down without any additional equipment, although the power convertor may cost you slightly more than one which cannot regenerate.

FEEDFORWARD FROM ANOTHER PROCESS VARIABLE

All we have looked at so far is speeding up the response to a change in setpoint by adding a rate-of-change of setpoint to the setpoint. However, there are many types of control system where response can be improved by adding a rate-of-change of some entirely separate variable.

For instance, on many centre-driven winders, reelers or coilers for paper or for metal strip, you need to control tension in the strip to make sure that you get a good roll or coil. If the tension must remain constant, then the torque you provide from the motor must increase as the diameter of the roll or coil builds up. You can design a system to calculate the diameter of the roll or coil, and to adjust the torque setpoint accordingly.

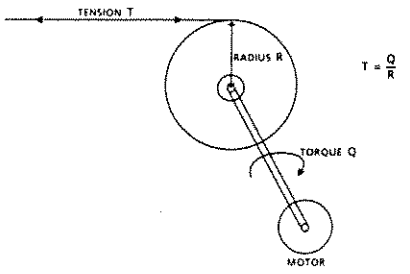

Fig. 78
Centre driven winder

However, if you now accelerate the line, you need to provide additional torque to accelerate the inertia of the roll or coil, and also the inertia of the driving motor. If you don't do this, the tension will drop during acceleration. In the worst case, it could drop to zero, in which case the strip becomes slack and may spill all over the place. On deceleration, the reverse will happen: the tension will rise and might even cause strip breakage.

To provide the acceleration/deceleration torque, we obviously need to add an extra signal to the torque setpoint. Further, since faster acceleration requires more accelerating torque, we must obviously take this signal from rate-of-change of line speed. Note that, on deceleration, the rate-of-change of line speed will be negative, and so the additional signal will then reduce the torque, which is exactly what we need.

The additional signal, generally referred to as "inertia compensation", also depends on coil diameter and strip width. The inertia which must be accelerated consists of a constant (for the motor and mandrel), plus a term proportional to the 4th power of diameter (for the coil). However, for a given strip speed, the motor speed will be inversely proportional to diameter. Consequently, the system for the setpoint and inertia compensation feedforward is as below:

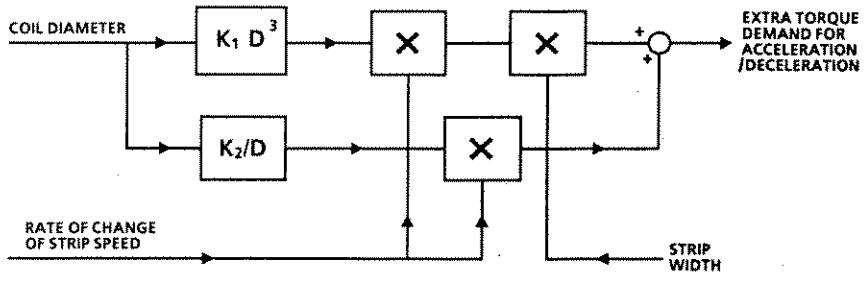

Fig. 79
Inertia compensation feedforward on a coiler

The above system may look rather complicated. However, if you don't use such a system, you may well find that the acceleration and deceleration rates you can use without running into tension problems are unacceptably slow.

In practice, you might have a system where you didn't have a torque control as such, but separate armature current and field current controls. In this case, you would need a slightly different inertia compensation system to what is shown in Fig. 79. It would still make use of rate-of-change of line speed and diameter, although the calculations would be slightly different. The resulting signal would add to the basic armature current setpoint to call for the additional acceleration or deceleration current.

SYSTEM WITH TWO MEASUREMENT TRANSDUCERS

We looked at this type of system in the last Item (Pages 102 and 103) as an example of a cascade control. In this example, you will remember that we fitted two temperature transducers. We put the first one as close as possible to the heat exchanger outlet. This gave us a temperature measurement with very little transport lag, and we used this to provide the measured value for the fast inner loop. The second transducer we put downstream at the place where we really wanted to control the temperature. We used this transducer to provide the measured value for the outer loop, where it was compared with our desired temperature setpoint.

Although such a system will respond quickly to disturbances, it will not respond so quickly to changes in setpoint. If we want a faster response to setpoint changes, then we can feed the setpoint forward to the inner loop as well, as in Fig. 80.

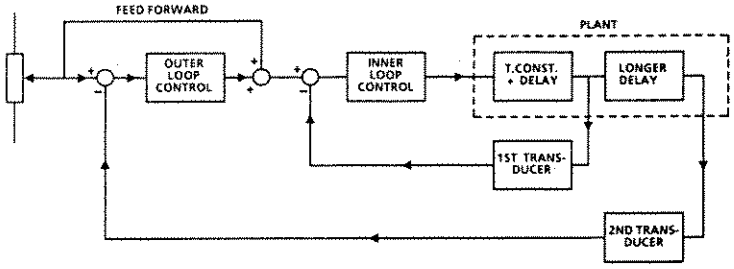

Fig. 80
Feeding the setpoint forward to the inner loop

When we do this, the overall setpoint for the inner loop is the sum of two signals - the real setpoint, plus a signal from the outer loop.

Now, the two temperature transducers will not measure exactly equal temperatures. If the fluid is hot compared with ambient temperature, then there will be some heat losses in the pipework between the first and second transducers. We therefore expect the first transducer to register a slightly higher temperature than the second. Consequently, the outer loop must provide a signal which calls for a slightly higher temperature from the inner loop than the real setpoint we have fed forward would on its own.

In fact, our outer loop becomes just a trimming loop. The setpoint we have fed forward calls for the basic temperature required from the heat exchanger, while the outer loop adds on a relatively small signal for the heat losses in the pipework.

Feeding the setpoint forward to the inner loop makes no difference to how we set the gains of the PID control on the outer loop as regards stability. However, the outer loop now has less work to do. It may only ever need to output a signal of, say, 10% of the setpoint under normal running conditions.

Besides normal running, we must also consider start up conditions. When we first wind the setpoint up, the outer loop (which, as we've remarked, is slow relative to the inner loop) could go into limit. If this limit were set to 100% of setpoint, and we applied 100% setpoint, the inner loop would get an overall setpoint of 200%. This wouldn't matter if, as the measured value rose, the outer loop reduced the setpoint back down again to, say, 10%. However, because it is slow, it might not do so in time before the temperature had seriously overshot the desired value.

We therefore need to set the limits down on the outer loop (or on any trimming loop, for that matter) so that we can't get excessive setpoints under any conditions. In many cases, we can set them to around 5 or 10%. This does, however, allow us to provide an overall setpoint of 105 or 110% to the inner loop. If we can't accept this, then we may need either to:

1. Use a LIMIT Special Function S32 on the overall setpoint before we feed it to the inner loop, or else:
2. Make the limits on the outer loop PID function variable, equal to 100% minus the real setpoint.

If we use the first method, we still need to set the limits of the PID function on the outer loop down to 10%, say, to prevent integral wind up (see Item 11, Page 62). With the second method, we only need one extra rung in our program.

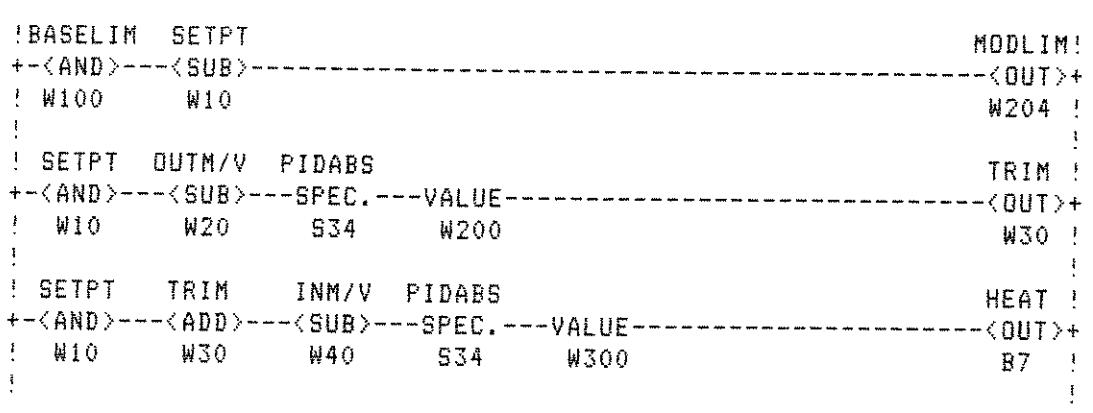

Fig. 81
Variation of trim loop limits with setpoint

In the above Fig., the second rung controls the outer loop and the third rung controls the inner loop. The output from the second rung (TRIM) is the additional signal adding to the setpoint SETPT in the third rung. The sum of the two is then compared with the measured value of the inner loop (INM/V), and the difference used to control the heat output via the second PIDABS function.

The first rung is our extra rung to make sure that the sum of the setpoint and trim signal in the third rung cannot exceed 100%. Since the positive limit of the PIDABS in the second rung is held

in table location Zm+4 (or in other words W204 in the example), then the first rung adjusts this value. The basic overall limit BASELIM is set to 100%, and thus the modified limit MODLIM in W204 will get set to 100% minus the setpoint. If the setpoint ever reaches 100% itself, then the output limit for the PIDABS in the second rung would reduce to zero, and remove the trim signal.

SUMMARY ON FEEDFORWARD

1. To get a faster system response to a change in setpoint, you can add onto the setpoint a rate-of-change of setpoint signal.

   Where the setpoint is ramped, and where you don't want the measured value either to change faster than the ramp or to overshoot, you can cancel the system time constant but not the system time delay.

   Where the setpoint can change as a step, the response of the measured value will not be able to precisely follow the setpoint. The rate at which the measured value can change will be limited by the available forcing on the plant and not by the control system.

2. For other types of system, you may need to speed up the response when some external process variable changes. To do this, you will need to use a rate-of-change of the external variable. You may need to first multiply it by appropriate functions of the system's own process variables before adding it onto the setpoint.

3. All feedforward signals are set up open loop. They normally provide exact compensation for one plant operating condition only. They work better over a wider range of operating conditions if the plant is linearized and/or if the strength of the conpensation signal is modified by multiplying it by (or by a function of) other variables.

4. For the particular case of a system with two process measurements of the same variable at different locations, providing the measured values for inner and outer loops in cascade, you may be able to feed the outer loop setpoint forward to the inner loop. The outer loop then becomes just a trimming loop.

   For any type of trimming loop, you need to make sure that you cannot provide too large an overall setpoint to the inner loop.

----oOo----

INTRODUCTION

In manufacturing industry, there are many applications where a number of individual controls receive the same setpoint, or a modified version of the same setpoint.

For instance, in a steel rolling mill for producing steel strip, there are usually several "stands" where the strip passes through rollers which squeeze the steel strip thinner. As a result, the strip leaving any stand will be travelling somewhat faster than the strip entering the particular stand. When you have a strip mill with several stands, then the rollers in any stand must be rotated slightly faster than those of the preceding stand. We therefore need a setpoint for the speed control of the electric motor driving any stand which is slightly higher than the setpoint to the preceding stand.

However, when you start the strip mill going, you need to accelerate all the stands together. You therefore need to have a common master setpoint for the whole mill, and then to modify this to generate each of the individual stand setpoints by multiplying it by the appropriate factors. You could do this as in Fig. 82 below.

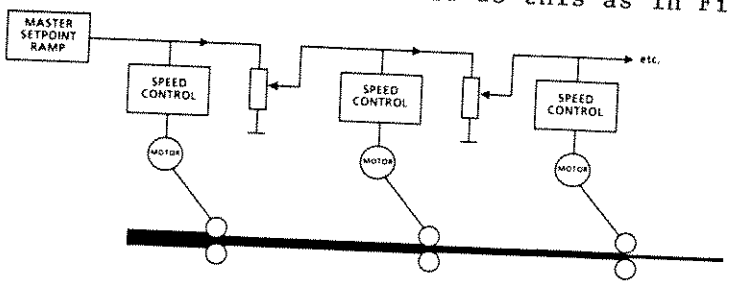

Fig. 82
Block diagram of setpoints for a strip mill

You get an exactly similar situation with paper, rubber and plastics lines where there needs to be some slight speed ratio or "draw" between different sections of the line.

RAMP GENERATORS

When you come to start up any of the types of process mentioned above, where the process contains a number of sections in a flow line, then you should start up all the sections in a co-ordinated manner. If you tried starting up the sections so that each one reached its operating condition in minimum time, regardless of what the other sections were doing, then:

- Some sections would reach their operating condition before others. As the process would not be ready to start until the slowest section had reached its operating condition, you would not save any time.

- You have used unnecessary peak power in starting up the process, since some drives have accelerated faster than necessary.

Further, on a plant like a paper or steel mill, you may be able to manufacture a product of saleable quality even during run up if you can maintain the correct speed ratios between sections or stands while accelerating.

It is therefore common practice to ramp up the setpoints to their working values, rather than altering them rapidly. With a GEM 80 controller, you can do this using the RAMP Special Function S33. You can then keep all the setpoints in step with each other, and you don't take unnecessary power accelerating any section faster than the others.

You could, of course, use a RAMP for each setpoint. However, it is usually simpler to derive the setpoints for a number of sections from a common master setpoint using a single RAMP Special Function.

RAMP SET TOO FAST

If you use a RAMP on a common master setpoint for a sectional system, you may still want to start the plant up as rapidly as power demand constraints will allow. However, as you try making the RAMP faster, sooner or later one of the sections will run up against any power constraint built into its individual control. While all the other sections follow the ramping master setpoint, this particular section will be forced to lag behind all the others. Depending on the settings of the required ratios between the individual setpoints in the line, it could be a different section which hits its power constraint first for different operating conditions of the plant.

For instance, on a strip rolling mill, each stand will have a speed regulator receiving its setpoint from the GEM 80 equipment. Each speed regulator will contain current limit circuits, to prevent the motor and the source of current (usually a thyristor convertor or inverter) being overloaded. Depending on how much you reduce the thickness of the strip on each of the stands, then the loadings of the various stands will not be exactly the same fraction of their current limit settings. It could therefore be a different stand which hits current limit first, when you try increasing the ramp rate, depending on what reduction you are taking on each stand.

To cater for this type of problem, the RAMP Special Function has data table location Zm+11 which you can use to reduce the ramp rate. Refer to the Software Data Sheet for the S33 RAMP Special Function, Publication T314. When you use a RAMP for ramping the setpoint to a single control loop, this location is not normally used, and its contents by default remain at zero. However, if you place a value into this location, you can slow the ramp down.

To make use of this location, first of all we need a signal from each section of the plant. Each of these signals must be analog or numeric, and must either be proportional to motor current or else proportional to current above a given level. We can then use these signals so that, as soon as any drive is about to reach current limit, the RAMP is slowed down.

When we do this, we have constructed a closed loop system. For instance, as soon as one of the current signals starts to slow the RAMP providing the setpoint down, then the motors will accelerate less fast, take less current, and slow the RAMP down less. In practice, if we have made the closed loop stable, the system will reach a steady state during acceleration.

How you handle the signals from the drives depends on the form these signals take.

- Some drives provide analog signals proportional to current. You then need to put in dead bands so that you only provide signals to the GEM 80 above a certain level. For each drive, you will need to set this level close to, but slightly below, current limit. You will then only get signals from the drives when they are near to current limit, but otherwise you will get zero signal.

- Some drives provide analog signals which are proportional to current above a certain level. That is, the deadband is built into the drive. This saves you having to provide any dead band circuits yourself.

- Some drives may provide digital (numeric) signals, equivalent to the analog ones described above, with or without dead bands built in.

Having got our current signals, we could simply add them together, and use the resulting signal to slow down the ramp. This would be O.K. if we could guarantee that we never got two or more drives trying to slow down the ramp at the same time. If we did, then our closed loop would suddenly have two or more times the loop gain which it would have had with only one drive trying to slow the ramp down. This could make it very difficult to stabilize the system.

Instead, it is preferable to just take the largest signal, from the drive nearest to current limit. You could do this either in the GEM 80 program or, in the case of analog signals, with external circuitry. Fig. 83 shows how you could do this with diodes and operational amplifiers.

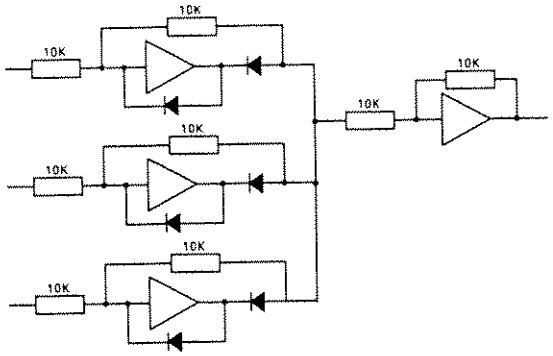

Fig. 83
Analog maximum amplitude selector

If, on the other hand, you have brought each analog signal into the GEM 80 by way of a separate analog input channel, then you can use the MAX Special Function T7 as below:

```
!DRIVE1     MAX      MAX                                                  !
+-<AND>---SPEC.---SPEC.-------------------------------------------<OUT>+
!  A10      T7       T7                                            G240  !
!        +-<    > +-<    >                                                !
!DRIVE2  !         !                                                     !
+-<AND>-+          !                                                     !
!  A11             !                                                     !
!                  !                                                     !
!DRIVE3             !                                                     !
+-<AND>----------+                                                       !
!  A12
```

Fig. 84
Maximum amplitude selection by program

With just a single signal coming from our maximum amplitude selector, we can now design our closed loop system. To make it clear how to analyse the system, we show the original system in Fig. 85, and then the system redrawn, taking just a single drive, in Fig. 86, where we have made the system look like a conventional closed loop block diagram.

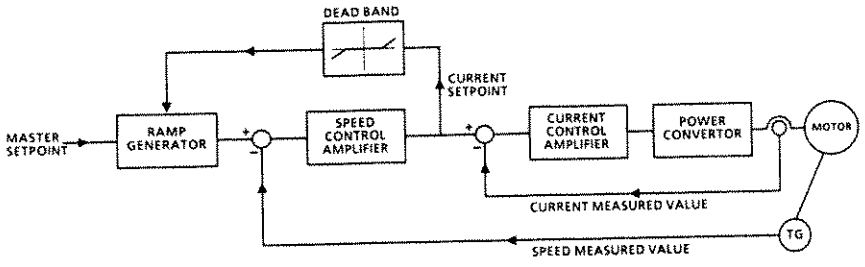

Fig. 85
Ramp rate reduction control system

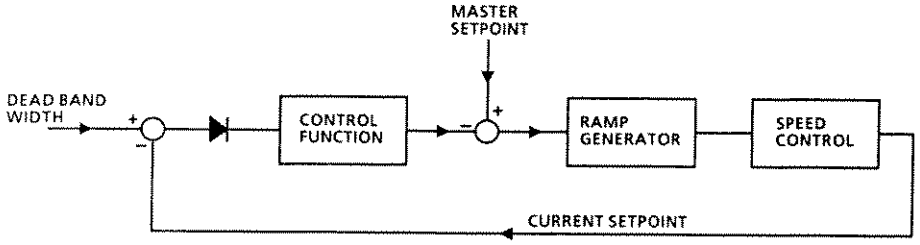

Fig. 86
Ramp rate reduction control redrawn

Effectively, the setpoint is the dead band which a current signal must exceed before we take any action to slow the ramp down. The measured value is the signal from the drive motor current. The error signal is therefore the amount by which the current exceeds the dead band.

Our plant consists of the ramp generator and the drive control. The normal rung input to the RAMP function (the master setpoint for the line) can be regarded as a disturbance. If this increases, then our error signal has got to increase to back it off, so as to keep the ramp rate constant and hence the accelerating current drawn by the motor constant at the dead band level.

Our closed loop control is, in fact, a spillover system (see Pages 103 and 104) which only operates when the current signal exceeds a certain level. When the signal is lower than this, then the error will be negative, and it will therefore get blocked and not affect the ramp generator.

In the "plant", the ramp generator is, as far as it affects the current signal, an integral term. In addition to this, we have the drive control system. What we need to know is how the drive system responds when we change the speed setpoint.

There are two possible cases:

1. For some applications, a drive may be restrained from accelerating by the back tension in the strip from other drives.
2. For other applications, the material will be sufficiently elastic for any extra current in the drive motor to cause it to accelerate independently of the other motors.

For Case 1, if we make a small change in speed setpoint to a single drive, then the extra current caused to flow through the motor makes little difference to the speed. If the speed control loop had just proportional control, then the change in current would be proportional to the change in speed setpoint. The rest of our "plant", the ramp generator, acts as an integrator. If we now put in just a gain between the current signal and the RAMP current limit input Zm+11, we will get a stable system as long as we don't make the gain too high.

However, most motor speed controllers are not just proportional, but include an integral term. They are, in fact, PI controls. The setting of the integral term is designed to make the speed loop have a good response. Consequently, we can't alter this setting just to make our ramp rate reduction control stable. Instead, we have to make the gain in the ramp loop give a response roughly equal in speed to that of the speed control itself. This means that the gain we put in must not be too low, as then we get a response which is oscillatory at a low frequency. (A system of this type, where the response goes unstable if the gain is either too low or too high, is referred to as a conditionally stable system).

For Case 2, if we make a small step change in setpoint to a single drive, then it will take accelerating current to bring the drive up to the new set speed but, after this, the current will decay back to its original value. However, because of the ramp generator, we must now integrate this current. We then get a change in signal proportional to the change in speed setpoint which behaves roughly like a time constant. The signals we get are as shown in Fig. 87.

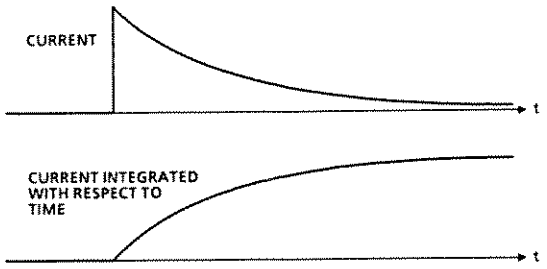

Fig. 87
Current and integrated current signals for a speed change

For a modern drive system, the effective time constant on the integrated current signal will be 50 ms or less. Any effective time delay will come primarily from the GEM 80 controller program cycle time, and is likely to be of the same order. We therefore have a system which is of the type we called Special Case 3 (see Item 8, Page 45). To stabilize it, we need to put in a long time constant or integral term, otherwise we will not be able to get any worthwhile gain round the loop.

The question is where. We could put the time constant or integral in one of three places:

- On the current measured value.
- On the difference between the current setpoint and measured value.
- On this difference but after the "diode" (see Fig. 86).

In fact, because the current setpoint is a fixed value, the first two options are equivalent to each other. Both of them delay the signal on the input side of the "diode" becoming positive, and

therefore produce a delay when you want to slow the ramp rate down. It is therefore better to use the third option when you want to start slowing the ramp down.

However, the reverse applies when you have finished wanting to slow the ramp down. When the current measured value drops below the current setpoint, the "diode" blocks the resulting negative signal, and therefore the output of the time constant just decays at its natural rate. We have no forcing to get the output down quickly (see Item 18, Page 109f). In fact, if we use an integral term rather than a time constant, the output doesn't decay at all but just stays there. On the other hand, we would get this forcing if we used one of the first two options.

To get the best of both worlds, what we need to do is to use the third option, but to switch out the diode action while there is any output from the time constant. This allows us to force down the output of the time constant or integral again when the difference between the current setpoint and current measured value is negative; we then reinstate the diode as soon as the output of the time constant or integral reaches zero. In ladder diagram terms, we can do this with a rung as follows:

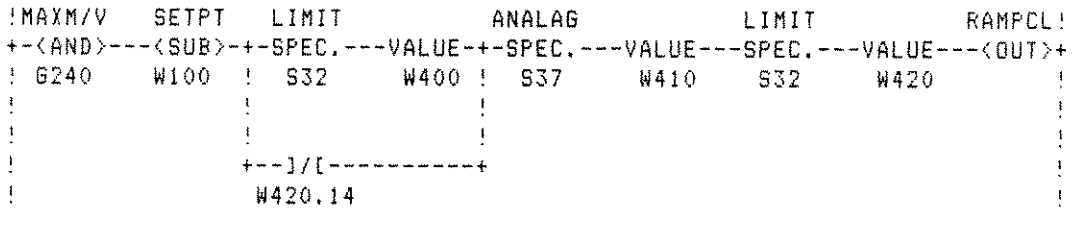

Fig. 88
Ramp rate reduction control program

In the above rung, both LIMIT Special Functions have table values set so that they can give zero or positive outputs only. That is, the Zm+1 values are 0 and the Zm+2 values are 32767 (see software data sheet for LIMIT), or W401 = W421 = 0, and W402 = W422 = 32767.

As long as the maximum current measured value shown as MAXM/V is less than the setpoint, then there is a negative number as the input to the first LIMIT, and therefore no output from it. If we assume that the contact W420.14 is open, then there is no input to the ANALAG Special Function and no output from the rung.

Now assume that one of the drives is taking current close to current limit, so that MAXM/V exceeds the setpoint. We now have a positive number as input to the first LIMIT, and the same positive number therefore passes through as input to the ANALAG. The output from the ANALAG will change more slowly but will also be positive, and will not therefore be limited by the second LIMIT. We therefore get some output from the rung immediately the MAXM/V exceeds the setpoint.

As soon as the output of the second LIMIT moves away from zero, then W420.14 changes from 1 to 0. This particular bit is set to 1 whenever the input to the LIMIT equals or is below the lower limit value (which we set to 0). Consequently, the N/C contact closes and bypasses the first LIMIT in the rung.

If the MAXM/V signal now drops below the setpoint again, the resulting negative number is not limited to zero but can pass straight on to the ANALAG. This will cause the output of the ANALAG to be forced downwards much faster than if its input were just zero. Once the output of the ANALAG passes through zero, then the second LIMIT prevents the output going negative, the flag bit W420.14 gets reset to 1, and the bypass contact opens again.

SUMMARY ON COMMON SETPOINTS

1. In a flow line process, use a ramp generator to raise the setpoints for all sections together. This way, you don't waste power accelerating any section faster than necessary.

2. If you set the ramp rate too fast, one or more sections in the process line could hit power limit, and would then lag behind the ramp. To prevent this, use a current or power signal from each drive, take the largest of these signals, and use it to slow the ramp down if it exceeds, say, 95% of the available power.

3. How you stabilize the resulting closed loop depends on the nature of the process.

   - If the material or strip is inelastic and prevents any individual drive accelerating independently of the others, then use a simple proportional control (as the ramp provides integral action).

   - If the material or strip is elastic so that each drive can accelerate independently, then use a time constant or an integral term. Once you get an output from your rate reduction program rung, switch out the "diode" action until the output is zero again.

----oOo----

INTRODUCTION

On many types of system, you may want to arrange for control to be switched from one operating mode to another (e.g. auto/manual), or for command to be transferred from one operator's control panel to another operator's panel.

Unless you design your GEM 80 program to properly cope with such changes of operating mode, then the plant variable you are controlling can hiccup when the change-over takes place. To avoid such hiccups, you should design your system to give "bumpless" transfer.

INTEGRATORS AND COUNTERS

We noted that, in a GEM 80 program, we could produce an integrator by constructing an up-down counter as in Fig. 34 (Page 50). The basic technique for bumpless transfer is to preset the value held by such a counter or an integrator so that, on switching from auto to manual or vice versa, or on switching from one operator's panel to another, this value already matches the value of the signal we have just switched from.

(By an integrator, we are not necessarily talking about the integral term of a PI or PID control. For bumpless transfer, we can also make use of the Special Function RAMP, S33 or, if it has a sufficiently long time constant, the Special Function ANALAG, S37.)

For instance, we might have a flow control where the valve was normally actuated by a closed loop control from a flow setpoint, but where the control could be switched over to manual control where the valve was actuated open loop from a different setpoint. In block diagram form, we might therefore have:

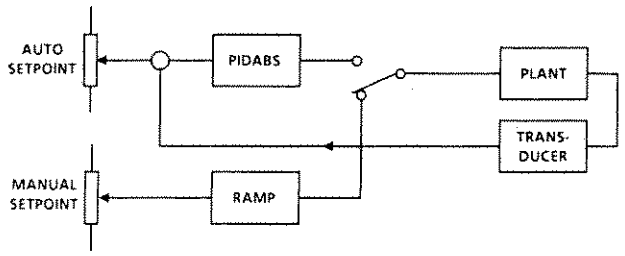

Fig. 89
Basic scheme for auto/manual change-over

The important thing to note is that, before the change-over switch, we have integral terms in both lines. For auto, it is provided by an integral term in the PI or PID control; for manual, it is provided by the ramp generator.

TRACKING

In order to make use of the integrators shown in Fig. 89, we need some method of presetting the output of the particular integral

which is not in use. If we are operating in auto mode, then we want to set up the output of the ramp generator always equal to that of the PI or PID control output, ready for a change-over to manual mode. If we are operating in manual mode, then we want to set up the output of the PI or PID control always equal to that of the ramp generator, ready for a change-over back to auto mode.

This idea of getting one value to keep in step with another is generally referred to as "tracking". It is a lot easier to do in a digital system than in analog circuitry. We can produce a GEM 80 program, equivalent to the scheme of Fig. 89 but with tracking added as below:

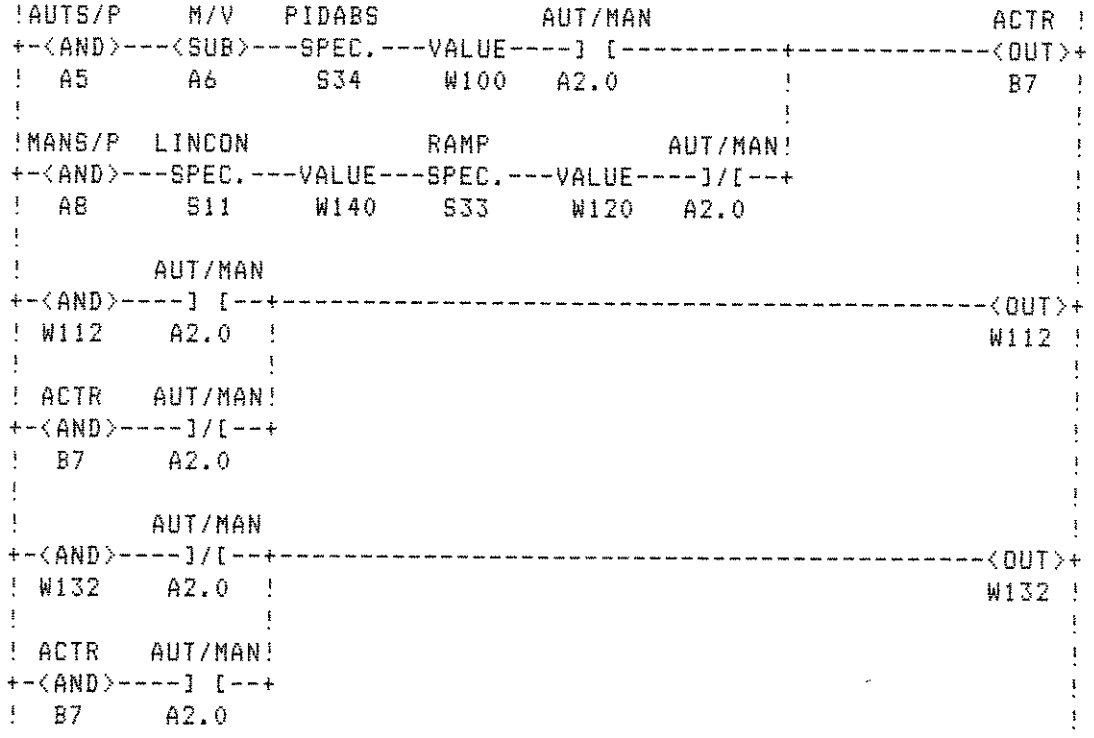

Fig. 90
Bumpless change-over using tracking

In this example, if the AUT/MAN input is ON, it gives the auto mode. In the top rung, the output B7 is the determined by the PIDABS Special Function operating on the difference between the auto setpoint A5 and the measured value A6. If A2.0 is set OFF, we are then in manual mode, and the output B7 is taken from the lower branch of this first rung, from the manual setpoint A8 via a LINCON to set any required gain and offset, via the RAMP Special Function.

The second and third rungs give the bumpless change-over. In auto mode, the second rung does nothing: it simply loads the value from W112 back into W112. However, the third rung loads W132 (the table location for the previous output of the RAMP in the first rung) with the value from B7, making it equal to the current

output signal to the actuator. If, on the next execution of the program, A2.0 was changed to OFF to transfer to manual mode, then the new output to B7 would be given by the RAMP starting from the previous value of B7, regardless of the setting of A8. B7 will eventually take up a new value dependent on the value of A8, but this may take several executions of the program depending on the ramp rate set for the RAMP Special Function. The change in B7 value is therefore smooth, and is not a sudden step.

In manual mode, the role of the rungs is reversed. The third rung now becomes a "do nothing" rung, putting the value from W132 back into W132. The second rung, however, loads W112 (the table location for the previous output of the PIDABS in the first rung) with the value from B7, making it equal to the current output signal to the actuator. If A2.0 is now changed to ON, the new output to B7 will be given by the PIDABS starting from the previous value of B7, regardless of the difference between the auto setpoint and the measured value at that time. Again, a smooth transition on change-over is obtained.

PRE-LOADING

Instead of using tracking and updating the value of W112 and W132 on every program execution, you could design your program so that you only updated the RAMP or PIDABS previous output table value on a change-over of A2.0, immediately prior to executing the control rung (i.e. the first rung in Fig. 90 above). This would shorten the average execution time of the program, except that for a program as simple as Fig. 90 the saving would be minimal. However, if you were doing a change-over of several controls at once, then there could be some benefit.

The program shown in Fig. 91, using the same control rung as in Fig. 90, uses this technique. It contains two BLOCKs, one of which is executed for a change-over from auto to manual, and the other for a change-over from manual to auto. When there is no change-over, then both BLOCKs are skipped.

The last rung is used to remember whether the previous execution of the program took place with the control in auto mode or in manual mode, by storing the previous state of A2.0 in G100.0. If A2.0 changes before the next program execution, A2.0 and G100.0 will be of opposite state (i.e. one of them will be ON and the other will be OFF) at the start of this program execution.

If the control changes from auto to manual, A2.0 will be OFF but G100.0 will be ON, and BLOCK 1 is executed. On a change from manual to auto, A2.0 will be ON but G100.0 will be OFF, and BLOCK 2 is executed. When the end of the program is reached, then G100.0 is once again set to agree with A2.0, so that neither BLOCK will operate on further program executions until there is another change-over.

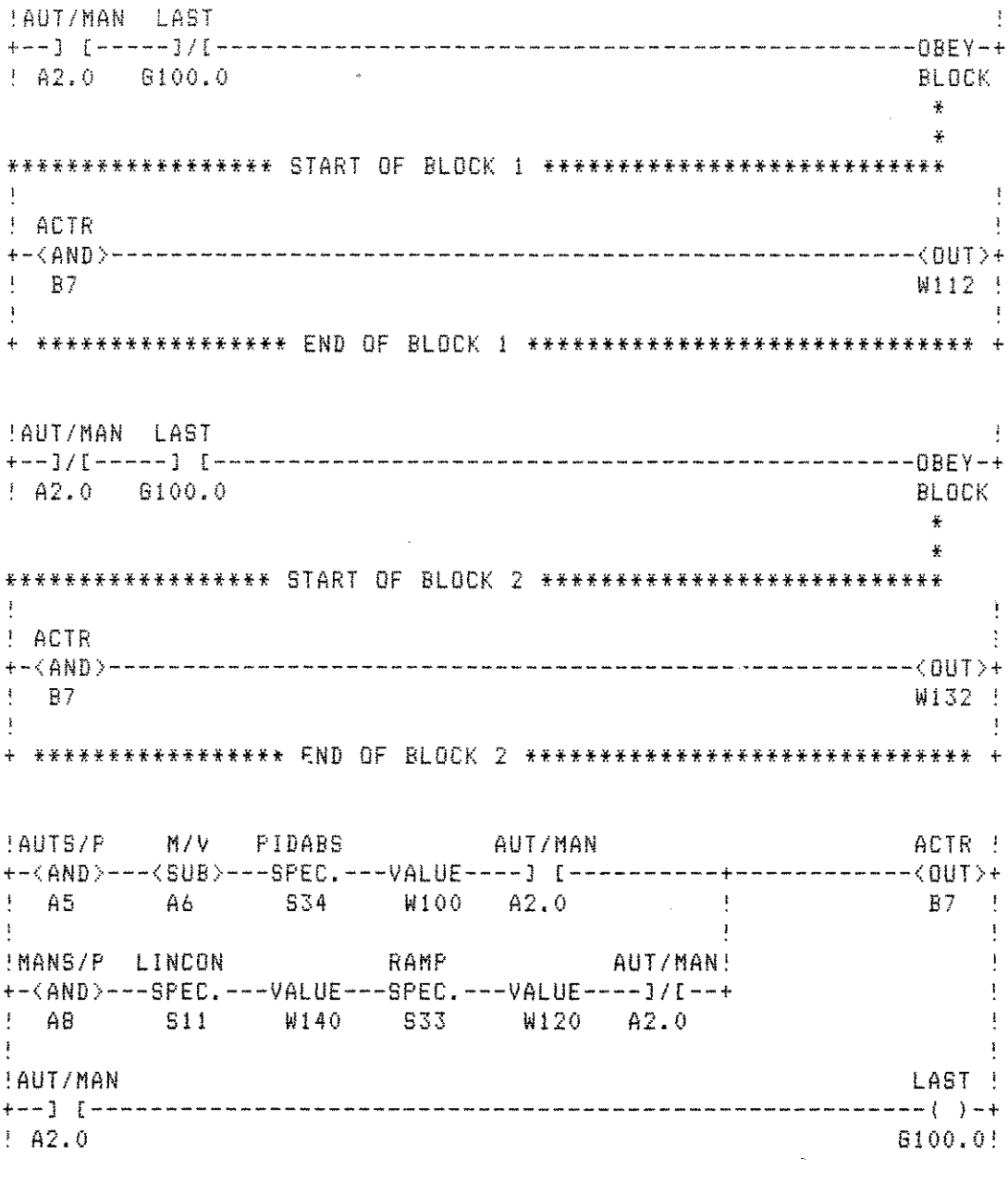

Fig. 91
Bumpless change-over using pre-loading

If you had to change several controls at once from one mode to another, then you could use a similar program with several control rungs. In this case, you could include several pre-loading rungs in each BLOCK, rather than just the single rungs shown in this example.

SUMMARY ON BUMPLESS TRANSFER

1. To get jerk-free transitions between two different modes of control of an output to the plant, you need to switch between

two rungs (or branches of a rung) where each contains an integral term or long time constant.

2. The integral term which is not controlling the output must be set to a value equal to the actual output, so that no change occurs on change-over.

3. Presetting the integral term can either be done by continuous updating (tracking of the output) or by preloading the output value on change-over immediately before use.

4. Although the resulting transitions will be jerk-free, there will still be a change in output if the setpoints do not match. See the next Item for transfers with no change at all.

----oOo----

INTRODUCTION

In the previous item, we succeeded in obtaining transfer from one mode of control to another without a kick to the value of the variable we were controlling. If there was any change following a transfer, it was always ramped. This gave us a smooth change but, nevertheless, a change.

There are some types of plant where you may want to change the mode of control, with no alteration of the variable being controlled on change-over, not even a ramped change. Typical of this is the transfer of control from one operator's panel to another.

You may also, however, want no change when switching from auto to manual or vice versa. Now, the only way in which this can be done is for setpoints, and not just outputs, to be set to their correct values.

IDENTICAL CONTROL MODES

Suppose you have a single control loop where the setpoint can be switched to come from one of two (or more) identical sources, such as operators' control panels. The way you need to handle the setpoints depends on the type of setpoint input device, which could be:

- Some form of keyboard.
- Raise/lower pushbuttons.
- Rotary or thumbwheel switches.

With a keyboard or pushbuttons, an operator cannot tell what the current value of setpoint is, so you must provide him with a setpoint value display. With input from different operators' panels, you must design your ladder diagram so that only one panel is in command at any instant. The setpoint value held in table in the GEM 80's memory should only be updated when keys or push-buttons are pressed on the particular panel selected to have command. If keys are pressed on a panel which does not have command, then your program must ignore this. To avoid any confusion, you should give the operator an indication whether or not he has command from the particular panel with which he is attempting to enter data.

With rotary or thumbwheel switches, the input you get from them is continuously present, and does not have to be remembered like the input from keyboards. If you transfer control from one operators' panel to a second, then there is no guarantee that the switches on the two panels will be set to the same value. The operator must therefore adjust the switch setting on the second panel to a setpoint value identical to (or within a small margin of) the setting of the switch on the first panel. You must therefore provide the operator with a display of the setpoint value so that he knows what to adjust the switch setting to. Further, your

program should be arranged so that the operator is only given command from the particular panel after he has adjusted the switch correctly, and equalized the switch settings.

Note that, if you are switching not just the source of setpoint but between identical but separate control loops, then you may also need to set up loop outputs to give bumpless transfer, using either the tracking or pre-loading techniques described previously.

DISSIMILAR CONTROL MODES

Where we want to switch control between loops which are not the same, with no alteration of the variable we are controlling, then we have got a new type of problem. Consider the case of wanting to switch between auto and manual control, as in Fig. 89, with no alteration of measured variable. Where the auto mode is the normal operating mode, we are hardly likely to bother about calibrating the manual control to any degree of accuracy, especially as, being an open loop control, it is liable to changes with disturbances like ambient temperature.

Suppose we had both the auto and manual setpoints set to 50% of span. Under auto mode, with a closed loop control, we could expect to maintain our plant variable accurately at 50% value, the accuracy depending mainly on that of the measurement transducer. However, when we change over to manual mode, with a roughly calibrated open loop control, our plant variable might change to a value quite a lot different from 50%. So, how can we set the manual setpoint to give us exactly 50% ?

Well, we already know a technique for keeping two values in line with each other which fits the bill. It's closed loop control. In this case, though, the closed loop we are going to use will be entirely within the GEM 80 program.

TRACKING BY CLOSED LOOP

For bumpless transfer, we previously dealt with a tracking technique where, under auto mode, we kept the output of the manual control (i.e. the output from the RAMP in the lower branch of the first rung in Fig. 90) in step with the output of the auto control (from the PIDABS 3-term control function) by loading this output value into the previous-output table location for the RAMP.

To arrange for a no-change transfer, we must still set the RAMP output to a value matching the PIDABS output, but we do this by controlling the manual setpoint. Conceptually, we do this as in the block diagram shown in Fig. 92.

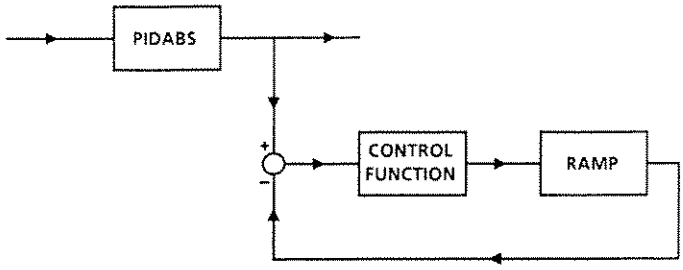

Fig. 92
Closed loop for tracking

What we have here is a closed loop where:

- The setpoint is the PIDABS output
- The measured value is the RAMP output
- The "plant" is the RAMP Special Function

The action of the closed loop will bring the RAMP output to a value equal to the PIDABS output. However, we are achieving this by altering the manual setpoint to the correct value. We therefore not only avoid bumps on change-over from auto to manual, but we also get no subsequent change in the plant variable we are controlling (until the operator deliberately alters the manual setpoint, or until there is a disturbance which the open loop manual control will not compensate for).

For change-over in the reverse direction, from manual to auto, we need a similar closed loop control where, in this case:

- the RAMP output is the setpoint
- the PIDABS output is the measured value
- the "plant" is the PIDABS function.

We have got a bit more design work to do, because we have got two extra closed loops, both of which need checking for accuracy and stability. However, let's look at a GEM 80 program incorporating these tracking loops assuming that they are simply proportional controls using LINCON Special Functions. This is shown in Fig. 93.

In this program, the auto setpoint AUT/S/P is held in W10 and the manual setpoint MAN/S/P is held in W20. We assume that these values are updated by a keypad in a portion of program not shown here (the design would depend on whether the keypad provided a serial port input or basic I/O input, etc.). We also assume that, while the system is in auto mode (when the AUT/MAN input A2.0 is energized), the operator is prevented from altering the manual setpoint and, while in manual mode (when the AUT/MAN input is de-energized), he cannot alter the auto setpoint.

The first three rungs are equivalent to the control rung (the first rung) in Fig. 90. By splitting it into three rungs, the intermediate auto output AUT/O/P and the manual output MAN/O/P are made available, as W160 and W180 respectively, for use in the

tracking rungs. While in auto mode, the fourth rung alters the manual setpoint held in W20 so as to get the manual output W180 to track the value of the auto output W160. In manual mode, the fifth rung makes the auto output W160 track the manual output W180 by altering the auto setpoint held in W20.

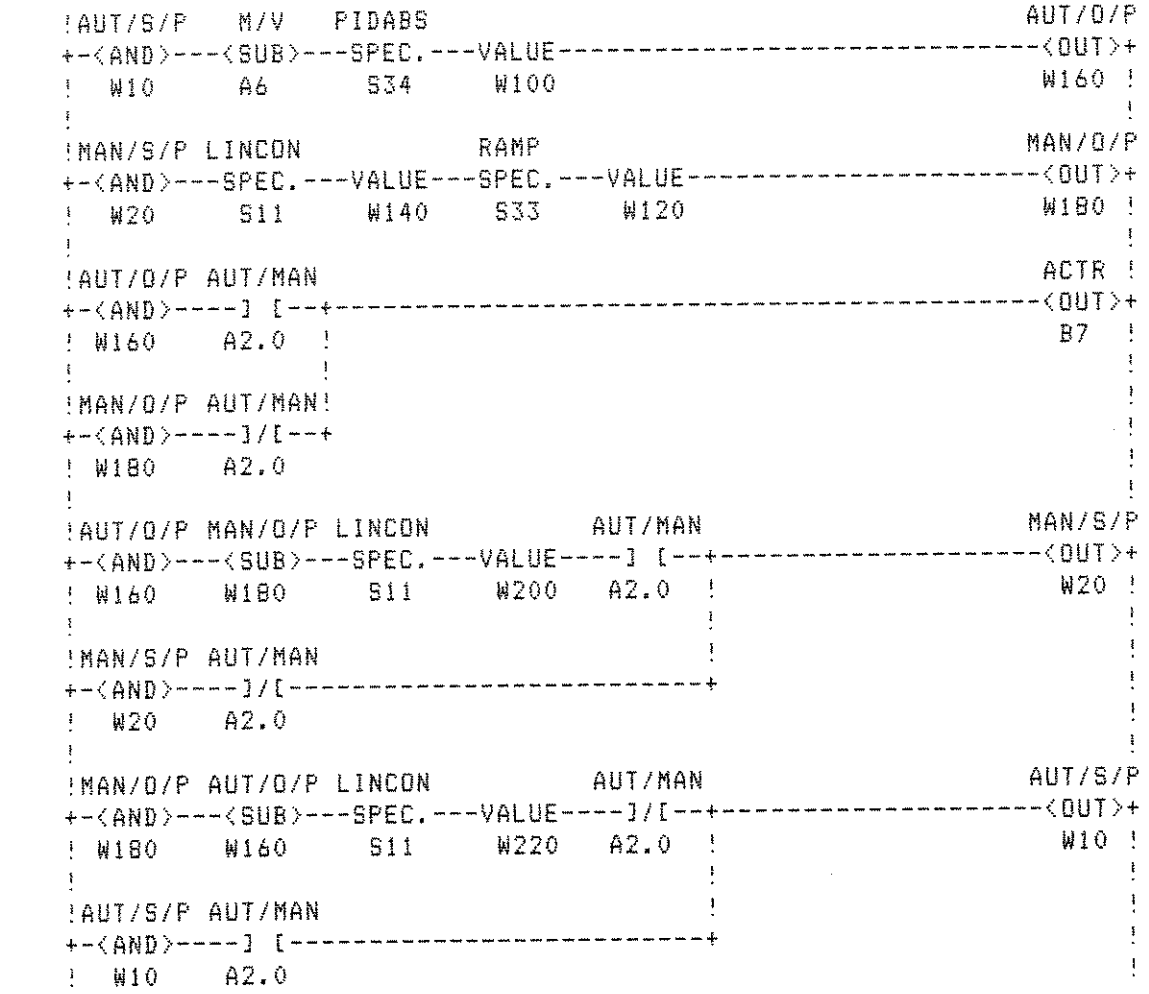

Fig. 93
Change-over using closed loop tracking

Within the GEM 80 program, in auto mode, there is therefore a closed loop consisting of the 2nd and 4th rungs. In manual mode, there is a closed loop consisting of the 1st and 5th rungs.

STABILITY OF PROGRAM INTERNAL CLOSED LOOPS

We now need to look at the accuracy and stability of these two closed loops, to find out whether the proportional controls we have shown in Fig. 93 with their LINCONs are satisfactory, or whether we need something more sophisticated.

Let's consider auto mode first, where the closed loop consisting of the 2nd and 4th rungs adjusts the manual setpoint W20. Let's redraw the 2nd and 4th rungs, showing just the parts of the rungs which are operative in auto mode, to make the system easier to follow, as in Fig. 94.

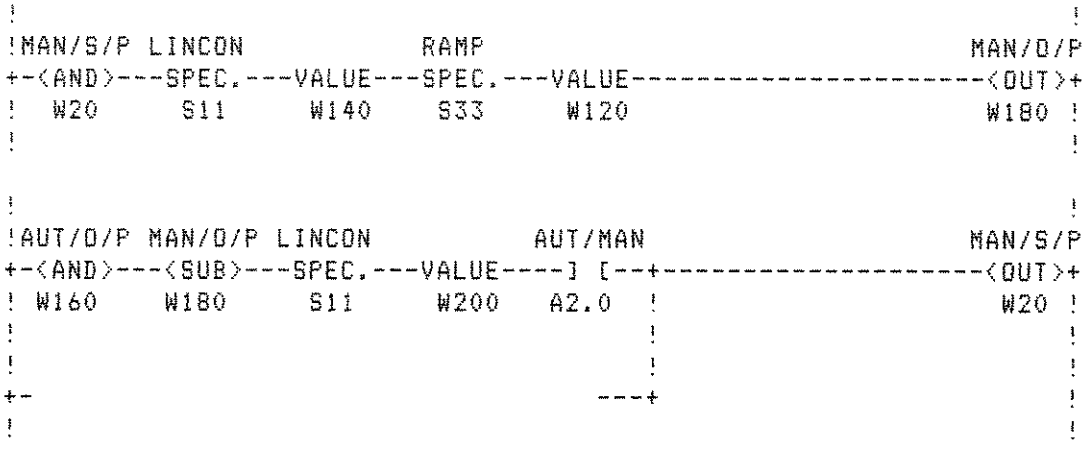

Fig. 94
Program closed loop rungs setting up the manual setpoint

Our normal approach to finding what we can do as regards gain and response is (as given in Section 1) to try running the system open loop and measure its open loop response. For this tracking loop, the setpoint is the auto output in W160, and our measured value is the manual output in W180 (see Fig. 92 block diagram). If we were actually to take a chart recording, then we would have to temporarily edit out the SUB| W180 to make the system open loop. Then, we would make a small change in the value of W160 and find out how W180 responded.

In this case, however, all our "plant" is contained in the first rung in Fig. 94. Knowing what values we have set up for the LINCON and the RAMP in this rung, we therefore have all the details about the "plant", so we don't actually need to obtain the results experimentally.

For open loop control, if we change the auto output W160 by 1 bit, then we want the manual output W180 also to change by 1 bit. Since a RAMP has unity gain in the steady state, we would therefore have to set the LINCON in the 2nd rung in Fig. 94, for open loop control, to have a gain which was the reciprocal of that of the LINCON in the 2nd rung. Thus, if the LINCON in the 2nd rung had a gain of 2, we would have to set the LINCON in the 4th rung to have a gain of 0.5 for open loop control. This would give the correct gain for open loop control, making the manual output excatly equal to the auto output.

If we found that, for closed loop control, we wanted a loop gain of M, then this would mean that (with the SUB| W180 in the first

rung reinstated) we would in this example set the LINCON to give a gain of M times 0.5. So let's check what M we can get.

To start with, the effective system time delay is one execution of the program. If you change the auto output W160 in the 2nd rung, it alters the manual setpoint W20 immediately, but this does not affect the manual output W180 (which is our "measured value") until the next program execution.

(Note: It makes no difference if the order of the rungs were reversed; by reversing the order of the 1st and 2nd rungs in Fig. 94, you could alter the contents of W180 immediately, but it would still be one execution later before it was compared with W160. It therefore still takes one program execution for the effect of a change to work its way right round the loop. With 2-rung closed loops internal to a GEM 80 program, it is a general rule that you will always get a 1-execution delay, regardless of the order of the 2 rungs.)

Next, what is the effective time constant of the system ? The only time dependent function is the RAMP Special Function in the 1st rung. Now, you might be fooled into thinking that a RAMP function was an integral term. It isn't. A proper integral term ramps up whenever there is <u>any</u> non-zero input. A ramp generator doesn't do that; it only ramps up until its output equals its input. Further, if the input signal to a ramp generator changes slowly, at a rate <u>slower than</u> the ramp rate, the output follows the input. Ramp generators only slow down the rate of change of the output, compared with the input, if the input changes <u>faster than</u> the ramp rate you have set. All the ramp generator does is to effectively act as a rate limit.

However, if you use rounding at the end of the ramp, the way that the RAMP Special Function provides this is to behave like a time constant (in a similar way to the ANALAG Special Function S37). The time response of a RAMP, for small signal changes, can therefore be more-or-less instantaneous if no rounding at the end of ramping is used, or it can be a time constant if rounding at the end of ramping has been programmed in the appropriate VALUE tables for the RAMP. We therefore have two cases to consider, depending on whether we are using rounding or not.

RAMP WITH NO ROUNDING

With no rounding, we have the type of system we referred to as "Special Case 2" in Section 1. That is, we have a time delay but negligible time constant. You may remember that, to stabilize such a system, we had to introduce a time constnt (see Page 45) or an integral term (see Page 77) into our program. The LINCON we have shown in fig. 93 and Fig. 94 after the SUB| W180 is therefore not adequate on its own. We therefore need to modify the second rung of Fig. 94 (i.e. the fourth rung of Fig. 93), maybe as given in Fig. 95.

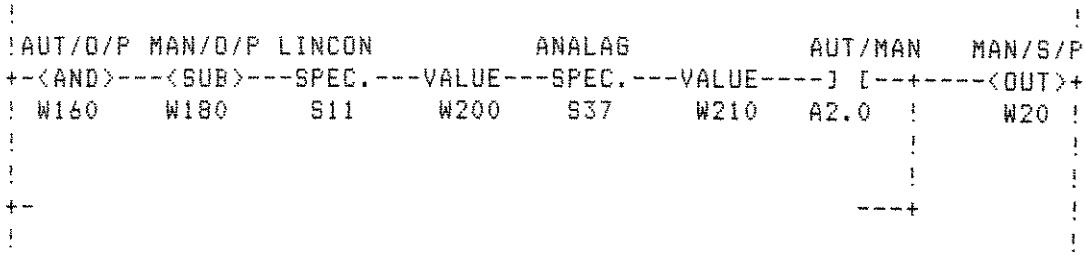

Fig. 95
Program closed loop rung modified for stability

We can make the time constant of the ANALAG as big as we like, but the longer we make it the larger we need to make the gain of the LINCON to give the best response. This will be roughly equal to a time delay of one progream execution plus a time constant of roughly equal time. Since everything we are dealing with is within the program, we can of course check how the loop performs without any plant inputs or outputs connected, varying the auto output AUT/O/P and seeing how well the manual output MAN/O/P follows it. We must, however, have the actual settings we are going to use already set up for the LINCON and RAMP in the first rung of Fig. 94.

RAMP WITH ROUNDING

If the RAMP in the first rung of Fig. 94 has been set up with rounding, then, when the difference between the rung input to the RAMP Special Function and its output is small, it behaves as a time constant. If this is large enough, we won't need to introduce any extra time constant as we had to in Fig. 95. If, however, it is fairly short, we may not be able to wind up the LINCON gain high enough to get a reasonable loop gain without putting in a much bigger time constant as in Fig. 95.

CONCLUSIONS

We don't want to stretch out this particular Item any further, as it covers only one small aspect of closed loop control. We haven't, for instance, covered how you stabilize the closed loop rungs for setting up the auto setpoint. The important point to remember is that, once you have finalized the settings for the first three rungs in Fig. 93 (or rungs in any similar system which control the plant), then your additional tracking rungs (rungs 4 and 5 in Fig. 93) are something you can play around with on your GEM 80 Controller without the plant connected, and experiment to your heart's content.

We also have not covered the use of integral terms in place of time constants in the tracking rungs. Obviously, integral terms are better because they make sure that there is zero error in the steady state. With integral terms, you need extra rungs to store their values by outputting to internal tables. You also need

to watch the placement in the rungs of the AUT/MAN contacts, otherwise your integral terms could continue counting up when you are not tracking. You could also put in feedforward to get one value to follow the other better under transient conditions when they are changing rapidly.

SUMMARY ON NO-CHANGE TRANSFER

1. To get transfer between two different modes of control of an output to the plant with no change at all, you need to keep the two output signals in step by altering the setpoints from which they are derived.

2. To do this, you may need to use closed tracking loops which are contained entirely within your program.

3. Where you construct a closed loop within a program from two rungs, you get a time delay of one program execution regardless of the order of the rungs. Any time constant or integral will, however, depend on the settings of the elements used in the rungs.

4. Once you have finalized the settings of the rungs controlling the plant, you can experiment with your tracking rungs off-line to get them set up for the best response.

----o0o----

INTRODUCTION

In some processes, there is some variable we would like to control which, for some reason or other, we can't measure. Maybe nobody has yet invented a suitable transducer, or maybe the quantity can't be measured at all.

A classic example of a quantity which can't be measured is the back e.m.f. of a d.c. motor. All you have are two armature terminals, across which you can measure a voltage. This voltage equals the back e.m.f., plus voltage drops in the brushes, armature windings, commutating poles, etc. There is no way in which you can actually measure the back e.m.f. on its own because all these other volt drops can't be separated out. So, back e.m.f. is something you can write down in the electrical equations for a motor, but try measuring it and it's no go.

ESTIMATION

If you can't measure what you want to control, then you obviously can't use a straightforward closed loop control. The first method of approach is therefore to estimate the output as accurately as possible, and to compare this estimate of the output with your setpoint, as though it were a genuine process measurement.

For instance, suppose you wish to control the mass flow on a conveyor belt. The mass flow will depend on the depth of material on the belt (i.e. the mass per unit length), multiplied by the speed of the conveyor belt itself. It may be difficult to measure the mass flow directly, but much easier to measure the mass of material on a given length of belt (e.g. from a belt weigher) and the speed of the belt (e.g. from a tachogenerator on the motor driving the belt). Hence, to estimate the mass flow, you could bring these signals back to the GEM 80 separately and multiply them together in your GEM 80 program. You could then use the resultant value as the measured value for your closed loop control.

As a second example, consider the back e.m.f. of a d.c. motor again. You could estimate this by measuring the terminal volts across the motor armature terminals, and then subtracting from this a signal proportional to motor current to allow for the resistive voltage drop. This means that you need a second measurement, of armature current. You could obtain this by a number of means, such as from the volt drop across a shunt, by using current transformers, by a Hall effect detector, etc. However, we have a problem with how big to make this signal. The resistance of the motor armature is not constant but varies with temperature. We might therefore need a third measurement, of motor temperature, so that we could vary the strength of the current signal. If we wanted to be more accurate still, we might want to compensate for the brush drop, and also for the transient volt drop across the armature's inductance when the value of the current changed. The more accurate you make your estimate, the more measurements you need, and the more complicated is the

setting up of the correct signal strengths. We are trying to set up a system not much different from that in Fig. 4, on Page 7, which we rather ridiculed at the time. However, if you really need to estimate something like back e.m.f., then you may be forced into a multiple measurement system.

MODIFIED SETPOINT

For some controls, in order to estimate a process variable we can't measure, we might have to multiply two variables together, as in Fig. 96:

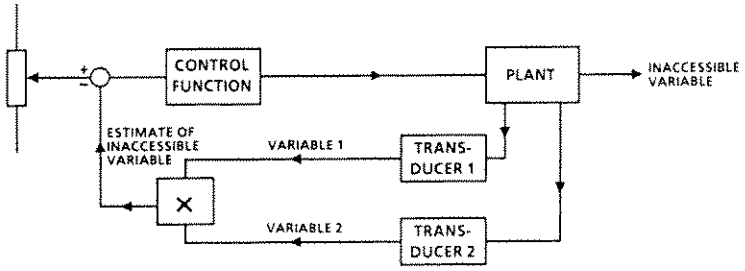

Fig. 96
Multiplying two process variables

Examples of where we might consider multiplying two process variables to obtain an estimate of measured value are:

- Torque control on a centre-driven coiler/reeler/winder, where you could find the motor torque by multiplying armature current by field strength.
- Fluid mass flow control, where you could multiply volume flow by specific gravity.
- Mass flow control on a conveyor belt system, where you could multiply mass per unit length by belt speed.

What we will now consider is whether the form of control shown in Fig. 96 is the best, or whether there may be a better alternative.

First, note that we have made measurements of two separate process variables, but we have only got one output signal from our Controller to the plant. Now, in spite of what some economists seem to think, you can't independently control two things at once with only one control signal. Only one of the variables will directly respond to our output signal. It is possible that some interaction between the two variables may occur, so that changing the variable we are controlling may cause a slight change to the other variable, but this will generally be a secondary effect that we can treat as a disturbance.

Second, because the secondary variable multiplies the first, we are varying the gain around the control loop for the controlled variable. In the worst case, if the secondary variable can ever be zero, then we get no loop gain at all. The system effectively becomes open loop, but with the control function gain much too high.

A better control strategy is often to use a control like Fig. 97:

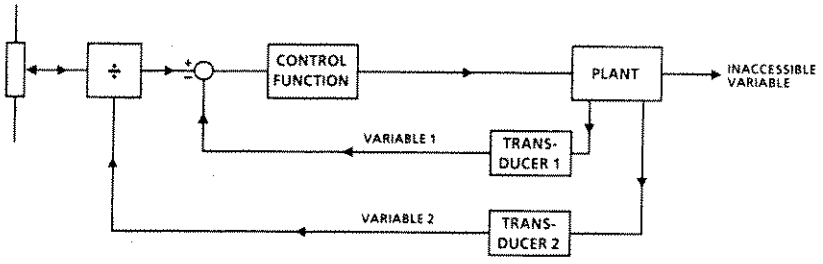

Fig. 97
Modified setpoint

Here, we have made our control loop directly control the first variable. The gain of this control loop doesn't vary with the second variable, as it did in Fig. 96, and so the control loop will be easier to stabilize. Now, to get the correct value of what we are trying to control for a given setpoint, we first divide the setpoint by the second variable. The two systems of Fig. 96 and Fig. 97 are, in fact, equivalent, even though the settings of the control functions would have to be different. If they include integral terms so that the measured values exactly equal the setpoints, then, in Fig. 96, we will have:

$$\text{Setpoint} = (\text{Variable 1}) \times (\text{Variable 2})$$

whereas, in Fig. 97, we will have:

$$\frac{\text{Setpoint}}{(\text{Variable 2})} = (\text{Variable 1})$$

which are mathematically equivalent.

Comparing the systems of Fig. 96 and Fig. 97, we can see that the system of Fig. 97 may often make it easier to stabilize the control loop, and therefore give us a fast response under all operating conditions. Note, however, that we are no longer estimating the quantity we want to control. Instead, we are controlling a subsidiary variable and modifying the setpoint.

For instance, if we want to control the torque from a motor, we cna estimate the torque by multiplying armature current and field flux, as in Fig. 96. If instead we use a Fig. 97 type of system, then we don't estimate the torque but instead control the armature current, with its own closed loop control. We then set the armature current setpoint by dividing the torque setpoint by field flux. Either way, we effectively control the same thing, but the Fig. 97 system is usually easier to stabilize.

One problem we had with the system of Fig. 96 was that, if the second variable fell to zero, we got zero loop gain. Now, we still have a similar problem with the system of Fig. 97. If the second variable falls to zero, then in theory the value of (Setpoint)/(Variable 2) becomes infinite. If we don't want this to happen, then we must either limit this value to a sensible maximum, or else limit the divisor to some minimum value.

SUMMARY ON INACCESSIBLE VARIABLES

1. Where a process variable you want to control can't be measured, then you will need two or more measurements from which you can infer its value.

2. You may then be able to produce a signal from the various measurements which is an estimate of the value of the process variable, and use this estimate as the measured value for your closed loop control.

3. Where you form an estimate signal by taking a basic measurement and adding to it a number of minor corrections signals, you will have a calibration problem. However, there may be no better method available.

4. Where you form an estimate signal by multiplying together two measurement signals, consider the alternative of controlling one of them and modifying the setpoint with the other - you may get a system which is easier to stabilize.

----o0o----

INTRODUCTION

There are a number of systems where two process variables both affect a third process variable. For instance, if you have a flow control, the flow will depend both on the valve opening and on the pressure. It is quite likely that, as you open the valve wider to increase the flow, the pressure upstream of it will tend to drop.

Now, your pressure might be set by a pressure control loop controlling a pump. As you open the valve to increase the flow, the pressure loop will therefore respond, increasing the pressure and hence the flow. The flow loop now responds by closing the valve to reduce the flow. This increase the pressure again. The pressure loop now drops the pressure, reducing the flow ... and so on. You've got an oscillation where two process variables (pump pressure and valve opening) both affect a third process variable (flow).

THE SOLUTION

To solve the problem, you must deliberately make one loop faster than the other. In the above example, the control loop which is meant to be controlling flow is the valve control. Suppose you make this loop much faster than the pressure loop. To be on the safe side to prevent interaction, you should make it 4 or more times faster.

With the system designed this way, then, when you need a change of flow, the flow control changes the valve opening until it gets the required flow. During this time, the pressure will drop, but the slower pressure loop will not correct it immediately. The pressure control then starts to raise the pressure but, as the flow loop is faster, the flow loop can adjust the valve opening as this is happening, and you don't get an interaction oscillation.

In theory, you could design the system the other way round. If you could make the pressure loop 4 times as fast as the valve control, then the pressure loop would always win, and hold the pressure rock steady despite what the valve was doing. In practice, it may not be possible to make the pressure loop fast enough for this, and you could end up with a very slow flow control loop. If this is the case, making the flow loop the faster loop is probably easier.

INACCESSIBLE VARIABLES

A similar type of interaction problem can occur with the type of system we dealt with in the last Item. Where there is a variable we can't measure, then we have to make two or more separate measurements, in order either to estimate the variable, or else to arrange some equivalent control like that shown in Fig. 97.

Now, it is quite likely that, for a system like Fig. 97, you would want to use closed loop control on <u>both</u> variables. You then have

a choice to make - which of the two variables do you treat as Variable 1, and which as Variable 2 ?

For instance, if you have a conveyor belt system where you want to control mass flow, then you could estimate mass flow by multiplying the mass per unit length (measured with a belt weighter) by the belt speed (measured with a tachogenerator on the driving motor). If, on the other hand, you use a system like Fig. 97, then you would probably use the belt weigher to provide the measured value to a closed loop onto the feeder depositing material onto the belt, and an entirely separate closed loop onto the motor driving the belt, using the tachogenerator to provide the speed measured value, as below:

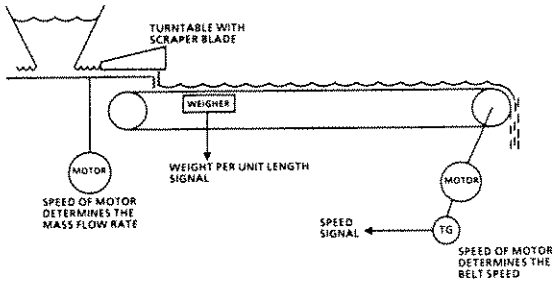

Fig. 98
Conveyor belt system

Given this arrangement, then you could control mass flow in one of two ways:

1. By taking the mass flow setpoint, dividing it by the belt weigher signal, and using the resultant signal as the setpoint for the speed loop.
2. By taking the mass flow setpoint, dividing it by belt speed, and using the resultant signal as the setpoint for the feeder loop.

Both of these arrangements are of the Fig. 97 type, but which is better ?

For Option 1 above, we rely on the feeder loop being fast enough to maintain the thickness of material deposited on the belt constant. If this loop is slow, then speeding up the belt will immediately cause the material to be spread more thinly, giving no change in mass flow. Further, because we are dividing the mass flow setpoint by a smaller signal from the belt weigher, the speed setpoint will get even bigger. Subsequently, because the belt weigher signal falls, the feeder loop will now start to deposit more material on the belt. Since the belt is moving too fast, we then get too high a mass flow. We could, in fact, get an oscillation just like the one we described on the previous page for flow/pressure controls.

If, on the other hand, we use Option 2, and hold the belt speed constant, then we should get our new mass flow as rapidly as the

feeder loop can respond. (This could be rather slow because of the transport lag between the feeder and the belt weigher). However, as long as the speed control is fast enough, and can quickly increase the motor power as the load on the belt demands, there should be no interaction.

Practicalities

In practice, you won't decide whether to use Option 1, Option 2 or some other variant just on the basis of which system gives the faster response, but you will have to consider economics as well. For instance, if you keep the belt speed nominally constant, you can probably drive it with a cheap squirrel cage induction motor without any speed control. You have then got to vary the mass flow by the depth of material on the belt. Since a squirrel cage motor is an open loop device, it won't hold its speed exactly constant in the presence of disturbances so that, if you want an accurate mass flow control, you will still need the tachogenerator to compensate for belt speed variaiton. If the variation is expected to be only a few percent, then you may be able to use a Fig. 96 type of system, as the loop gain won't vary significantly.

Whatever system you use, however, it is important to check that you won't get interaction with other variables. Otherwise, you will have to deliberately make the response of one variable slower than the response of another just to make the system stable.

SUMMARY ON INTERACTION BETWEEN LOOPS

1. Where a process variable is affected by two independent controls, you can get interaction unless you make one control about four time faster than the other.

2. Where you have to make two independent measurements because a variable you want to control is inaccessible, check that the variables measured don't interact. Otherwise, you may have to adopt the same technique of making the control for one of them faster than the other.

----oOo----

INTRODUCTION

In Section 1, we dealt with dither oscillations in Item 13. We found that, using digital rather than analog control, we could get an oscillation where the measured value varied up and down by one or more ADC steps, and the output to the plant by a somewhat larger amount. If you wanted to eliminate the dither, we suggested that you put in a dead band. This eliminated the dither, but lost you some accuracy.

DEGAINING FOR SMALL ERRORS

There is, however, a technique you can use where you are particularly concerned about maintaining the system accuracy and eliminating dither, both at the same time. This is to drop the gain of the proportional and integral terms for small differences between the setpoint and measured value.

We therefore need to form this difference in a separate rung, prior to going into our PID control function, so that we can adjust the gains of the P and I terms first. We don't, however, need to adjust the derivative gain.

In Item 13, we worked out the size of dead band we needed for a proportional control, and we called this ±Y/2. When we are using degaining, we need to make the proportional gain proportional to the error when the error is within the band ±Y/2. We can do this with a LINCON preceded by a LIMIT, as below:

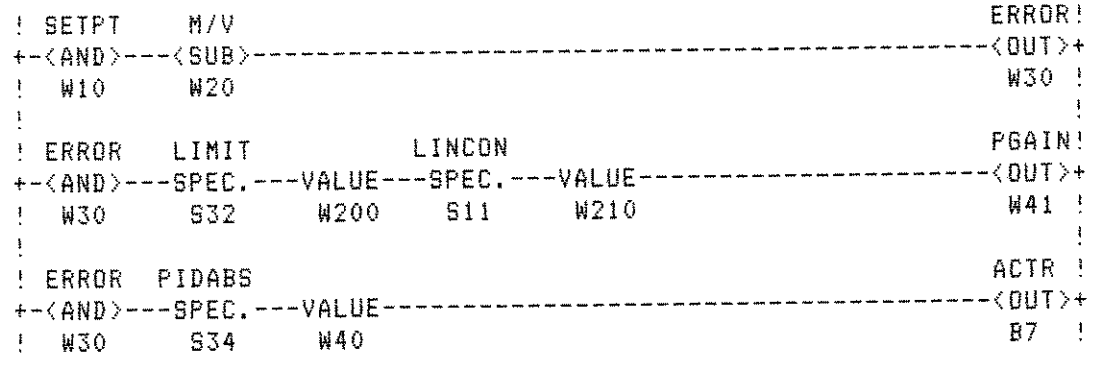

Fig. 99
Varying the proportional gain with error

Suppose that Y/2 = 14. Suppose also that we know that, for errors larger than this, we need a proportional gain of 20. We therefore set the LINCON gainsto give us a proportional gain value of 20 for an input of 14. The table value Zm+1 (i.e. W41 in the example) setting the proportional gain in a PIDABS Special Function is 100 times the actual value, so we must make the LINCON gain 2000/14. Now, for any error larger than 14, we will get a gain of 20, but for any error less than this the gain will reduce, e.g. it will be only 10 when the error is 7.

What about the integral gain ? Well, you may remember that integral gain needs to vary as the square of the proportional gain (see the top of Page 57). Once we start degaining the proportional term, then we must simultaneously degain the integral term but proportionally to the square of the error. If we don't do this, then the system will become less stable at low errors. We therefore need another rung for the integral gain for the PIDABS Special Function held in Zm+2 (i.e. W42 in our case). In this case we need to square the error by multiplying it by itself, and then we need a LINCON as for the proportional gain, but with a different setting. To save putting in the LIMIT twice or more, we can put it in the first rung, producing the output we have called LMTDERR (for limited error), as below:

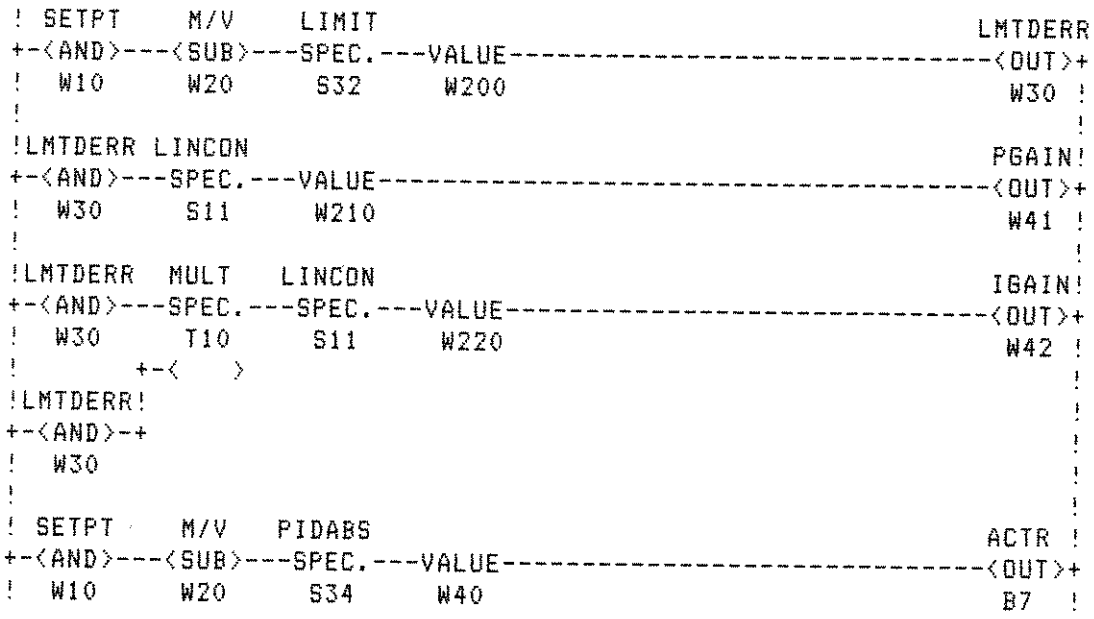

Fig. 100
Varying the proportional and integral gains with error

(Note that, in the third rung, we have squared the limited error LMTDERR by simply using the MULT Special Function. You can only do this if the rung inputs to the MULT function are less than 181, else the maximum positive number allowed of 32767 will be exceeded. In our case, where the LIMIT Special Function in the first rung is set to 14, there is no problem. If you had a system with a limit set higher than 181, then you would have to use LINCON instead of MULT with a suitable size denominator to keep the rung output small enough).

Now, the technique shown in Fig. 100 will theoretically give a control with no dither. But will it always work, and what will its response be like ?

RESPONSE

First, the response to a step change in setpoint may seem a little disappointing. The initial part of the response looks no different to a control with no adaptation of the PID control. The final part of the response (where the measured value gradually gets reset equal to the setpoint) is slower, because of the reduced gain for small errors. When the two do become equal, however, then there will be - in most cases - no subsequent dither.

Also, if your measured value is slightly noisy, tending to vary randomly by a few bits (by less than $\pm Y/2$), then the system will not try to correct the resulting small errors so rapidly, giving a more placid system. Any large error, however, will get corrected at the normal speed for a PID control.

CASES WHERE SETPOINT AND MEASURED VALUE CAN'T MATCH

Second, the system may not eliminate dither entirely if there are particular values of setpoint at which the measured value cannot become equal to it. This is a problem of quantization of the measured value, coupled with scaling and linearization.

For instance, if our measured value comes from an analog input module with an ADC which can only give values in the range ± 2047 in 1-bit increments, we might well scale this up by a factor of 10 in our program, to give numbers in the range ± 20470. However, this scaled up value can now only change in steps of 10. On the other hand, our setpoint may well be able to take on any number in that range.

If our setpoint were set, say, to 10005, then the best that our measured value could do would be to dither, alternating between the values 10000 and 10010. Consequently, the control scheme shown in Fig. 100 would not entirely eliminate dither, even though it might considerably reduce its amplitude.

The solution to the problem is to manipulate the setpoint so that it can only ever take on the same values as the measured value can take. In this instance, we must divide the setpoint by 10 and then multiply it by 10. The two effects do not cancel out because the intermediate result produced by the division, in integer arithmetic, can only vary in 1-bit increments. Multiplying by 10 then gives a number which varies in 10-bit increments. We can then always get a setpoint and measured value which match, and, combined with a Fig. 100 type of system, we have eliminated dither.

It does not matter whether you divide and multiply by an integer, as in the example we have just worked through, or whether you use two LINCONs to divide and multiply by a fractional number, e.g. 10.42, as long as the division gives a number which varies in increments of 1-bit.

Where we have more difficulty eliminating dither is where the measured value has been formed by linearizing the signal from an analog input module with an FGEN Special Function. In this case, the increments of the measured value are not regular over its span. In theory, you could manipulate the setpoint by feeding it through an inverse function followed by a function identical to that used for the measured value. In practice, because of the way an FGEN works, you cannot produce an exact inverse function (at least, not without using hundreds or thousands of break points). You can therefore eliminate dither, but maybe at the loss of some accuracy in the setpoint in its manipulation.

It may in such cases be simpler to linearize the measured value on the analog side, before it is converted to a number by the ADC, although this may lose you some accuracy in the measured value (see Item 2, Page 12).

CONCLUSIONS

The final question is whether eliminating dither is worthwhile, compared with allowing the system to dither, when you want an accurate PID control and therefore can't accept a dead band. This obviously depends on:

- How frequently disturbances occur.
- How clean and noise-free your process measurement signal is.

If disturbances are frequent or your process measurement is noisy, then you are going to get variations in measured value which may be larger and at a higher frequency than what you would get from dither. Any dither oscillations would then tend to be masked, and you might not notice any difference in performance if you eliminated them.

SUMMARY ON ACCURATE PID CONTROLS THAT DON'T DITHER

1. Instead of using a dead band, you can degain the system for small errors. This allows the measured value to exactly match the setpoint in the steady state without dither.
2. For stability, the integral gain must be reduced as the square of the proportional gain.
3. If the measured value can only move in steps which are either greater than 1 bit or are not regular, then you need to manipulate the setpoint so that it can only take on the same set of values as the measured value.

----oOo----

SECTION 3

POSITION CONTROLS

CONTENTS

(continued)

ITEM No. CONTENTS (continued) PAGE

----oOo----

POSITION CONTROLS

GEM 80 systems are often required to provide position controls. In a typical application, you will provide a position setpoint as a figure obtained from within the GEM 80 program or from values set up in tables. The measured position will typically be obtained from a shaft encoder or by counting pulses from a pulse generator. By subtracting the measured value from the setpoint, you can obtain a position error. You then need to manipulate this position error so as to provide an output to the controlling actuator, servo drive, or whatever, as in the block diagram below:

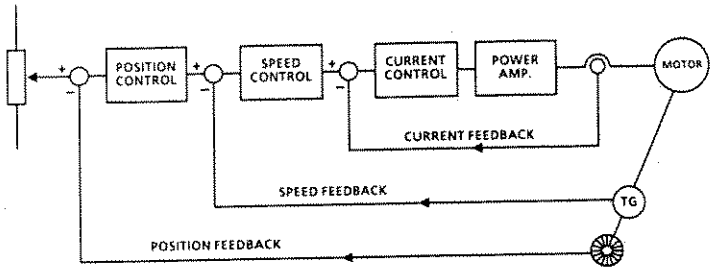

Fig. 101
Typical position control

The Items which follow give some practical guidance on the way in which you should design position controls, and on the effects which different position measuring devices have on the system design.

You may wonder why we have decided to devote a complete Section of this manual just to position controls. What is special about them, which requires any different design methods to what we have already developed in Sections 1 and 2 ? The answer is that we generally have some special constraints with position controls which force us into using non-linear control.

To begin with, we generally want a position control not to overshoot, whereas for most types of process control some overshoot is acceptable. Secondly, we generally have a constraint on the rate-of-change of the rate-of-change of position, or, in mathematical terms, the second derivative of position.

The rate-of-change of position is simply speed, and there is usually a limit on how fast the drive can be allowed to run (e.g. there is a maximum safe speed for an electric motor). The second derivative is the rate-of-change of speed, or the acceleration/deceleration of the drive, and, with a position control, this is usually limited as well (e.g. because of the maximum torque which a motor can provide).

If you ever want to control some variable entirely different to position where the second derivative must be limited, then you may be able to use some of the same design techniques given in Item 28 and Item 29 in this Section.

INTRODUCTION

A position control is a system containing an integral (what we called our Special Case 5, Page 47). As long as the motor is turning, the position increases. The only way in which you could run such a system open loop would be to set the speed setpoint feeding the drive control to a given value for a calculated period of time, and to hope that the drive moved the correct amount. If our system has to perform several movements, then each movement could introduce a further movement error. Since errors would be cumulative, we could soon finish up where our actual position bore little relation to the desired position.

LIMIT SWITCH CONTROL

If a position control has got to run to only a small number of preset positions, you could use limit switches to switch the speed setpoint off when the drive was a short distance away from a required position. Once you switch the speed setpoint off, then the drive will have to declerate to zero speed and stop. The question we have to ask is therefore: How far will the drive move after a limit switch is operated and the speed setpoint becomes zero ?

Clearly, we can't expect this stopping distance to be absolutely constant. Consider possible disturbances:

- Speed. When a limit switch operates, will the drive speed always be exactly the same ?
- Friction. Will any decelerating torque provided by friction in slides, screws and gearboxes always be exactly the same ?
- Motor braking torque. Will the drive system produce a braking torque when its setpoint is set to zero, and will this always be the same ? (This may depend both on the drive controller and on the motor characteristics).
- Mechanical brakes. Have you got brakes providing a braking torque, or are any brakes just used for holding position after the system has come to rest ?
- Load. Is there a load (e.g. on a hoist) which is assisting or hindering deceleration ? If so, will it vary ?
- Inertia. Even if a load is not providing an accelerating torque, is there a load of variable mass or inertia which needs to be decelerated, and which therefore requires a variable braking torque from the motor to slow it down ?

Assuming that your position control application has only a few preset positions to run to, and a limit switch control might be feasible, this is where you have to sit down, do a few sums, and find out what the variation in stopping distance is likely to be.

TWO-STAGE LIMIT SWITCH CONTROL

For a position control with a few preset positions, then, if your calculations show you that the stopping distance is much more variable than the stopping accuracy you require, one possibility is to use two limit switches in place of each one considered previously. The first one, instead of switching the speed setpoint to zero, switches it down to a creep speed value (e.g. 10%). This limit switch is sometimes referred to as the 'slowdown' limit switch. Then, the second limit switch, often referred to as the 'final' limit switch, operates and takes the speed setpoint to zero.

Stopping from 10% speed, rather than 100% speed, will improve accuracy roughly 100 times (i.e. for a constant deceleration rate, Newton's laws show that the stopping distance is proportional to the initial speed squared). However, you have still got to make certain that the drive has finished decelerating to creep speed before the final limit switch operates, for all conditions of the possible disturbances considered earlier. If the drive is running faster than creep speed when the final limit switch operates, it is very likely to overshoot the required position.

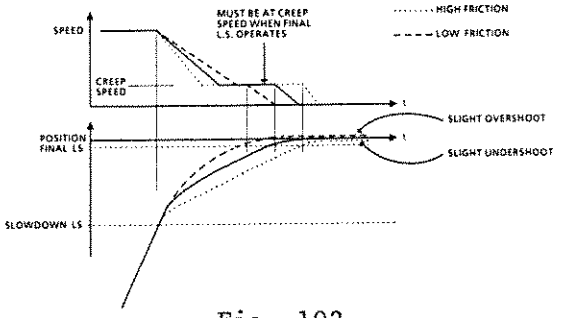

Fig. 102
Two-stage limit switch control

To cater for the worst case, you may have to place the slowdown limit switch further away from the final limit switch than you would need for normal operation. The drive will then spend some noticeable time running at creep speed. You have then achieved the accuracy you want, but at the expense of increased time, as the drive will waste time at creep speed before hitting the final limit switch.

Again, by doing some calculations, you can calculate what distance you need between the slowdown and final limit switches, and how much time will be wasted while running at creep speed.

CONTROLS WITH CONTINUOUSLY VARIABLE SETPOINTS

Obviously, limit switch operated position controls are only practical where you need the drive to run to only a few preset positions. If your application is one where you must be able to run the drive to any position, with a setpoint which can be any number within a given range, then limit switch controls are out.

You may then need to consider closed loop control, with a position measurement transducer.

STEPPER MOTOR CONTROL

For fractional horsepower, however, you may be able to use stepper motors. A stepper motor is one where a number of coils can be energized to pull a permanent magnet rotor to a finite number of different angles. To achieve continuous running, the stepper motor drive must energize the coils in turn. To move the rotor through a large angle (e.g. several revolutions), then the drive must energize the various coils a given number of times, and, for a given speed, at a defined frequency. It is then possible to turn the motor by a given angle from where it was previously, without needing any position feedback.

There are two possible problems, however. First, since a stepper motor only moves by a given angle from its starting position, you have got to know what this starting position is when you power up the equipment. This generally requires a zeroing procedure, where you run the motor (maybe at low speed) to a position mechanically defined by a limit switch or mechanical stop. Once you know where the drive is, then you can move it the required number of steps to any desired position.

Second, the stepper motor must be capable of providing sufficient torque to never stall or pull out of step. With insufficient torque, the motor could get out of position by a number of revolutions but then carry on running satisfactorily, although remaining out of position. You would then have to repeat the zeroing procedure before you could control position again. However, if you choose a stepper motor which can provide sufficient torque for abnormal conditions, then it will tend to produce pulsating torques for normal running (especially at low speed). For fractional horsepower and servo applications, pulsating torques may not be a problem but, on larger machines, such pulsating torques may not be acceptable mechanically (because of excessive gearbox wear, amplification of peak torques by mechanical resonance causing shaft breakage, etc.).

Consequently, stepper motor controls are usually restricted to fairly low power applications. If your application will require a conventional motor, and if limit switch control is not feasible (either because you have a continously variable setpoint or because limit switch stopping is not accurate enough or too slow), then you need a closed loop control with a position measuring transducer providing a process measurement of position.

SUMMARY ON OPEN LOOP POSITION CONTROLS

1. If your drive has only a few preset positions to run to, you may be able to use limit switch control. Check, however, that you will get the accuracy you need in spite of variations of speed, friction, braking torque, load and inertia.

2. If, again, your drive has only a few preset positions to run to, and single limit switch control will not give you the required accuracy, then you may be able to use pairs of limit switches, where the first limit switch reduces the speed setpoint to a creep speed value, and the second limit switch takes the setpoint to zero.

3. If your drive must be able to run to any programmed position, then you may be able to use stepper motors for low powers.

4. For higher powers, or where you can't accept pulsating torques, use closed loop control.

----o0o----

INTRODUCTION

If you have decided that you need closed loop control, then your first task is to look at the process measurement. We dealt with what to look for with process measurements in Section 1, Items 2 and 3. You might like to refresh your memory by referring to the Summaries on Pages 14 and 19.

There are quite a wide range of types of position measuring device - rotary and linear, analog and digital, etc. We give below some of the advantages and disadvantages of the various types.

ROTARY POSITION TRANSDUCERS

To obtain a measurement of position, you can use a transducer of one of two types: (a) incremental, or (b) absolute.

Incremental transducers only tell you how far the drive has moved from a datum position, and this datum has to be set up every time the equipment is powered up. Absolute transducers avoid this by giving a true position signal.

Incremental Rotary Position Transducers

The commonest form of transducer of this type is the pulse generator. The GEM 80 equipment and the GEM 80 program have to be arranged to count the pulses to determine how far the drive has moved from its datum position.

(Note: Some people refer to pulse generators as shaft encoders. This term is better left for devices which produce a code as the output signal, telling you what the position is as an absolute position number).

To set the datum, you must run the drive to a given physical position, and set the pulse count to zero within your GEM 80 program. In some cases, you can do this by running the drive against a mechanical end stop, probably at low speed. Where this is not feasible, you will need an input to the GEM 80, separate from the pulse generator, specially for datum setting. This input will generally come from some form of limit switch. For repeatability, you should use a non-mechanical switch (e.g. a magnetic or optical proximity switch) for preference.

The pulse generator itself must provide sufficient pulses per revolution to meet accuracy and resolution requirements (see Page 157 for the difference between 'accuracy' and 'resolution'). You don't, however, want a pulse generator producing an unnecessarily high number of pulses per revolution. Since the GEM 80 will only scan counter input modules at periodic intervals, then when the drive is running at top speed and the pulse generator consequently producing pulses at maximum frequency, you might find that the counter in the counter input module got full before the next program scan. You might then be forced to use a counter module with a bigger counter.

A second factor in the choice of a pulse generator is direction detection. If you only count pulses, you have no indication of whether the drive is running forwards or backwards. Even if your drive is only meant to turn in one direction, it might overshoot when approaching position. It is therefore essential that you choose a pulse generator of the type producing bi-phase pulse trains, i.e. one producing two pulse train outputs which are 90^{o} displaced. You can then determine direction from which of the two pulse trains is leading and which is lagging.

The remaining factors in the choice of a pulse generator which you need to consider are things like operating life, reliability, etc.

Absolute Rotary Position Transducers

We can subdivide these transducers into two types: those which give a digital output, and those that give an analog output.

Digital

Transducers producing direct digital outputs are normally referred to as 'shaft encoders'. Because they produce a binary number or code, you don't need any datum setting procedure on power up. Datum setting is a once-and-for-all procedure, of either mechanically rotating the encoder shaft or body relative to the motor, or else putting an appropriate value into P-table which is subtracted from the position feedback to make it zero at the datum position. On any power down and power up, the datum will remain set.

Although shaft encoders are readily available which give direct binary numbers as output, such numbers are only scanned by the GEM 80 at periodic intervals. Further, when a number is read, it cannot be guaranteed that all the bits which make up the number are read at exactly the same time, even though the difference in time may be fractions of a microsecond. If the GEM 80 scanned the shaft encoder output just as the number was changing, then it is just possible that it could mis-read the number (i.e. it could read some bits of the previous value and some bits of the new value). Although such a mis-read might occur only once in a blue moon, it is preferable to prevent this happening at all. You can achieve this by using a shaft encoder which gives an output in a Gray code (where there is only a change of a single bit between any two adjacent codes). You then need to convert from Gray code to binary within your GEM 80 program.

The remaining factors in the choice of a shaft encoder which you need to consider are things like operating life, reliability, etc.

Analog

One disadvantage of shaft encoders is that they need several wires in the cabling to them - one for each bit making up the Gray coded binary number, and two supply wires. Also, the signal levels tend

to be low (e.g. TTL logic levels), which can, depending on the installation, make them prone to electrical interference.

For these reasons, some system designers prefer to use resolvers. A resolver only needs 6 wires, but needs an appropriate input module to convert the a.c. signal levels into a binary number which can then be manipulated by the GEM 80 program. Because this conversion depends on the ratio of two signals, and not on their absolute amplitudes, a high accuracy can be achieved. Further, for the same accuracy, resolvers are usually cheaper than shaft encoders.

If you choose to use a brushless resolver, make sure that you excite it with a sinusoidal waveform. Square wave excitation normally only works satisfactorily with resolvers with sliprings and brushes.

Gearboxes/Turns Counting

With either a shaft encoder or a resolver, if it is rotated more than one revolution, then the output recycles. It follows exactly the same pattern as on the first revolution. To avoid this problem, you can either:

- Gear down the encoder or resolver, so that it makes less than 1 revolution for full travel of the motion.

- Count the revolutions turned.

If you want to use gearing, you may be able to find units which are supplied with internal gearboxes. Otherwise, you may need to obtain a precision anti-backlash gearbox to couple the encoder or resolver to the motor. In either case, the gear ratio must be such that full travel of the drive causes the transducer to rotate by less than 1 revolution.

The alternative to using a gearbox is to count turns, so that the encoder or resolver tells you where precisely you are within any revolution, but you only know which particular revolution you are in from the turns count. This makes the position measurement partly absolute and partly incremental. You will need to have a turns-count zeroing procedure, similar to that for datum setting with pulse generators, except it need not be so accurate, as you have only got to move the drive to within the correct revolution for datum setting.

With resolvers, there is a limit to the accuracy you can get from a single resolver (see manufacturers' data sheets). If you want a completely absolute position measurement and you don't want to count turns, then you could use two resolvers in a coarse/fine configuration. To do this, you must gear down the coarse resolver so that it makes less than one revolution for full travel, and gear down the fine resolver so that it makes several turns for full travel (usually 64 or 128 revs per coarse resolver rev.).

Coupling the Transducer to the Motor Shaft

You should couple the position transducer direct to the motor shaft, and not to what the motor is driving, unless you can guarantee that there is extremely little backlash. If you've got lousy mechanics in your equipment, don't expect the control system to compensate for them. Putting the position transducer where there is backlash between it and the motor can cause position control instability.

Even more important for stability is the mounting of any tachogenerator used within the drive speed control. Any backlash between the motor and the tachogenerator can cause worse problems than backlash between the motor and position transducer. Use couplings which are torsionally stiff; preferably use a motor with an integral tacho.

LINEAR POSITION TRANSDUCERS

For each type of rotary position transducer, there is generally an equivalent linear transducer formed by notionally flattening out the rotary one.

Incremental Linear Position Transducers

You can get linear pulse generators. Problems are, however, that whereas a rotary pulse generator is a self-contained unit protected reasonably well against the environment, it can be difficult to protect a long accurately divided strip against the environment. If you have got a number of applications with different length slides, then you can't carry a single type of spare.

Absolute Linear Position Transducers

A linear transducer giving a direct digital output is generally expensive and, again, difficult to environmentally protect. More common is the equivalent of a flattened out resolver which, being magnetic, is less prone to environmental problems.

ACCURACY AND RESOLUTION

When a position control comes to rest, then it should stop with zero positional error, as determined by the difference between the setpoint and the position measurement. For zero error, the position could then be anywhere within one bit:

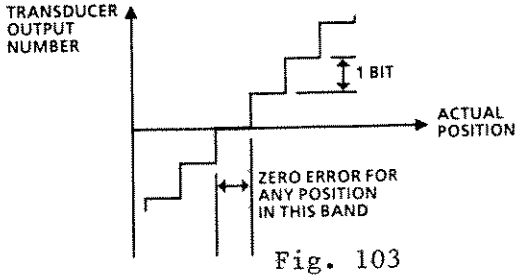

Fig. 103
Transducer accuracy

Your first consideration in choosing a position transducer is therefore that one bit represents a movement somewhat less than your required stopping accuracy.

For some types of position control, however, the setpoint does not change from one value to another value at distinct times, but may vary continuously, so that the position has to continuously track the setpoint. In such cases, you may need to use a position transducer where one bit corresponds to a much smaller movement than what you require for accuracy. For instance, if you need an accuracy of 1 cm, all you need is a position transducer which provides a change of 1 bit in the position feedback number seen by your GEM 80 program for a 1 cm movement. However, if you have a slowly changing position setpoint, changing at 1 cm/s say, then the drive would, every second, step on by 1 cm instead of moving continuously. If you don't want this to happen but you want the drive to move smoothly in a slow, continuous motion, then you need a position transducer with a much finer resolution (e.g. giving a change of 1 bit in the actual position number in the GEM 80 for 1 mm or less, even though you only need a 1 cm accuracy). Obviously, you also need to calculate the setpoints within your GEM 80 program to the same resolution in such cases.

TURNS COUNTING FROM ENCODERS BY PROGRAM

When you are using an absolute shaft encoder for the position transducer, but allowing it to make several revolutions (see Page 156), it may be possible to count turns within the program rather than by hardware. To be able to do this, you need to check out what the top speed of the encoder will be and whether the GEM 80 program cycle time will be quick enough to scan the encoder at least 4 times per revolution.

For example, if the encoder is direct coupled to a motor whose top speed is 1500 rev/min., which equals 25 rev/s, then one revolution at this top speed will take 1/25th of a second, or 40 ms. Therefore turns counting by program will only be possible in this example if you can guarantee that the GEM 80 program cycle will be less than 10 ms. If the program cycle is likely to be longer than this, then you would have to do turns counting by hardware.

Counting turns is just the same in principle as counting pulses from a pulse generator, although at a slower rate. That is, you need two pulse trains which are 90° displaced so that you can tell whether to count up or to count down, depending on the direction of rotation. With the majority of Gray code encoders, you will find that the most significant bit (MSB) is ON for angles between 180° and 360°, while the next most significant bit (NMSB) is ON for angles between 90° and 270° - just what is needed. The two signals then vary as in Fig. 104.

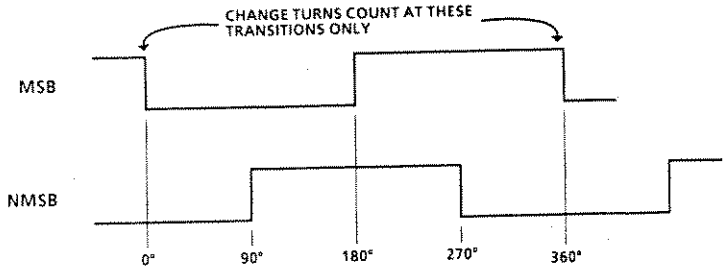

Fig. 104
Turns counting from a Gray-code encoder

If the encoder is turning forwards, then the condition for adding one to the turns count is that the MSB changes from 1 to 0 while, at the same time, the NMSB is 0. For the encoder turning backwards, the condition for subtracting one from the turns count is that the MSB changes from 0 to 1 while, again, the NMSB is 0. Note that, because we only take note of changes in the MSB while the NMSB is 0, we cut out taking notice of any changes from 0 to 1 or vice versa of the MSB halfway through a revolution (at 180°).

Suppose that the MSB is an input A7.11 and the NMSB is on A7.10, and that the count is held in W100. Then a possible section of program is as follows:

```
!        LASTVAL NEWVAL   NMSB    COUNT                          COUNT!
+-(AND)----] [-----]/[--+--]/[----(ADD)-----------------------(OUT)+
!   1    W101.0   A7.11 ! A7.10   W100                           W100 !
!                       !                                             !
!        LASTVAL NEWVAL !                                             !
+-(AND)----]/[-----] [--+                                             !
!  -1    W101.0   A7.11                                              !
!                                                                     !
!NEWVAL                                                          LASTVAL
+--] [-------------------------------------------------------------( )-+
! A7.11                                                          W101.0!
```

Fig. 105
Turns counting program

(Note that the AND -1 in the second branch of the first rung is superfluous, since the left hand vertical supplies -1 without any instruction being needed; however, the readability of the program is improved by leaving it in).

In the example section of program above, the count has been put in W-table, which would retain the value on power down. However, power down often occurs during mechanical maintenance, when the equipment may be man-handled out of position, causing the datum to be lost. It is therefore advisable to go through a datum resetting routine after power down as standard practice (see Item 31).

W101.0 forms a 'one-shot' such that only one count is made per transition.

----o0o----

ELECTRIC MOTOR DRIVEN CONTROLS

We will deal in this Item with electric motor driven position controls. In the last Item in this Section, we give some notes on differences between hydraulic position controls and electric motor driven ones.

SPEED CONSTRAINTS

Most electric motors have a maximum rated speed, and a corresponding maximum rated voltage. These may be limited by:

- The motor controller
- Mechanical strength of the motor
- Commutation (applies to d.c. motors only)
- Insulation strength

Motor Controller

With some types of motor controller, the output voltage to the motor is limited by some circuitry within the controller. Speed controllers using tachogenerators for the speed measurement may limit the maximum speed, and hence the maximum voltage. If the type of motor controller you are using has neither type of limit built in, then you must limit the maximum speed within your position control system to make sure you don't get overvolts or overspeed. With a motor speed controller, you just need to limit the maximum signal you are feeding to it. With a motor current controller, however, you must arrange to limit the maximum speed within your position control system.

Mechanical strength

Mechanically, motors have a maximum safe rotational speed. If you exceed this by too large a margin, the motor could burst because of centrifugal forces.

Commutation

Electrically, d.c. motors will only commutate without sparking if the voltage per commutator segment is below a certain figure. If you put too many volts on a d.c. motor so that you exceed this figure, then, besides causing overspeed, you will cause sparking. If this gets too bad, sparking may bridge between brushes and short the motor across. You then get a flashover, which may seriously damage the motor.

Insulation strength

If your motor is an a.c. one without a commutator (e.g. squirrel cage induction motor), then you won't get the above problem. Even

so, too many volts may not be good for any type of motor if design insulation levels are exceeded.

ACCELERATION/DECELERATION CONSTRAINTS

A most important factor in the design of position controls is constraints we must cater for on acceleration and deceleration rates. Four common sources of constraint are:

- Motor controller - the maximum torque it can provide.
- Stopping accuracy - too rapid deceleration may cause us to lose out on accuracy.
- Duty cycle - if the motor is continually accelerating and decelerating, it may overheat.
- Mechanics - the mechanical system may not stand the stresses imposed by too rapid acceleration and decleration.

Motor Controller

For most types of electric motor speed or current controllers, there are built-in limits to restrict the maximum accelerating and braking currents to values which both the motor and the power electronics can cope with. Since the torque produced by a motor is generally proportional to the current through it, then these current limits also limit the torque output from the motor, and hence the acceleration and deceleration rates of the motor.

However, note that, for a given motor current, the acceleration and deceleration rates depend on what the motor is driving as well as on the motor itself. If one equipment has a high inertia reflected to the motor shaft, whereas a second equipment with an identical motor and identical motor controller has a low inertia, the second motor will accelerate much faster for the same current.

If you can obtain details of the inertias, gearbox ratios, etc., of what the motor is driving, you may be able to calculate maximum acceleration and deceleration rates. If you can't obtain these details, you may be able to measure them using a chart recorder. If this isn't possible, then you may have to set up your position control by cut-and-try methods.

Stopping Accuracy

Any microprocessor based controller (whether a dedicated position servo module, or a GEM 80 Controller) can only alter its output signal to the motor controller at discrete time intervals. The only difference is that a dedicated module can generally do it more frequently as it has fewer other tasks to perform, so that the time intervals are shorter. As the drive approaches position, the position controller must obviously reduce the speed setpoint until, when the drive has reached the required position, the speed setpoint must simultaneously reach zero.

Since we are dealing with a digital system, the way in which the speed setpoint reduces will be as a staircase, not as a smooth ramp. The very last step in the staircase will switch the setpoint from a low speed value to zero. The distance that the drive moves on this last step is therefore just like that on a limit switch control. If the speed is too high when we switch the speed setpoint to zero, then the drive will overshoot the intended position. Then, on the next program cycle, the controller will have to output a reverse polarity speed setpoint to bring the drive back to position. There is even a possibility that the drive will never reach the required position, but will hunt about it.

To make certain that this doesn't happen, you may need to arrange for the setpoint to ramp down at a sufficiently slow rate to achieve the accuracy you need. You will need to set the step size so that, on the last step in this ramp, the speed setpoint prior to switching to zero will be sufficiently low for the drive not to subsequently move further than the required accuracy.

If you find that this accuracy constraint gives rise to very slow acceleration/deceleration rates, then you probably should not be using your main GEM 80 program to provide the position control, but you should be using a dedicated module instead, as this will give you much shorter time intervals between steps.

Duty Cycle Constraints

Where a speed control has built-in current limit circuits, the limit value is normally higher than the value of current which the motor can continuously carry. If, in your particular application, the motor is going to be spending most of its life accelerating and decelerating, then the r.m.s. value of the current could be too high, and the motor might overheat.

Note that the heating effect depends on I^2t, where I is the current and t is how long the current is applied for. If you halve the current, then, to accelerate to the same speed, you will have to apply this current for twice as long. However, the heating effect overall will be halved. See Fig. 106 below:

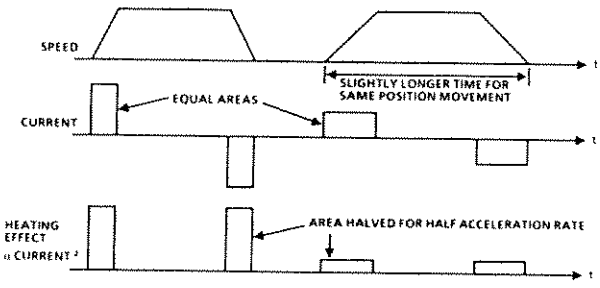

Fig. 106
Reducing r.m.s. current by reduced acceleration

You can reduce the maximum acceleration and deceleration rates by the way you design your position control. You will have to set these rates to lower values than you would obtain by simply allowing the drive to accelerate and decelerate in current limit. Obviously, if you do this, then the response of the position control when moving from one value of setpoint to a second value will be slower.

Mechanical Constraints

In some applications, it is not desirable to accelerate and decelerate at the maximum possible rates as this would cause faster wearing out of the mechanical system (e.g. screws and gearboxes). For such applications, you might need to limit acceleration and deceleration rates within the design of the position control, in the same way that you might have to with an r.m.s. current constraint.

SUMMARY ON DESIGN CONSTRAINTS

1. To design a position control, you need to have figures available for:
   - maximum speed you can use.
   - maximum acceleration and deceleration rates you can use.

2. Usually, the maximum speed you can use is just the speed given on the nameplate of the motor. (It could be less than this if the speed controller can't provide sufficient volts to get the motor up to its rated speed.) The other constraints on speed we listed in the text will then be automatically satisfied.

3. On the other hand, which of the possible acceleration/deceleration constraints is the limiting one is not necessarily obvious in any particular application. It could be set by:
   - the motor controller.
   - stopping accuracy.
   - duty cycle.
   - mechanics.

   To find the maximum acceleration and deceleration rates you can use, you will almost certainly have to do a few calculations.

4. If the constraint for stopping accuracy limits the deceleration rate to a significantly lower value than any of the other deceleration constraints, then you need a faster program cycle time. This may force you to use a dedicated position control module, rather than incorporating the position control into your main GEM 80 program.

----o0o----

INTRODUCTION

We saw in the previous Item that, when we come to design our position control, we have got to somehow limit the top speed and the acceleration/deceleration rates (that is, the rate of change of the speed).

Even if there are built-in constraints in the motor controller, so that it limits the top speed and the rate of change of speed, we still need to set limits on the speed or current setpoint we are outputting from the position controller. Remember what we wrote in the Summary on Limits at the end of Item 11 in Section 1:

> "Set limits and rate limits so that you provide a signal to the plant which is never larger or changing faster than the actuator or other controlling device on the plant can respond to."

In this case, our controlling device is the speed or current controller, together with the motor.

IDEAL RESPONSE

Before considering what a real position control can actually do, let us look at what we would like it to do, i.e. what the ideal response would be. Fig. 107 shows two cases. The first is for a short position movement, where the drive never gets up to top speed before it has to slow down again. The second is for a long position movement, where the drive reaches top speed, and runs for a time at this top speed before decelerating.

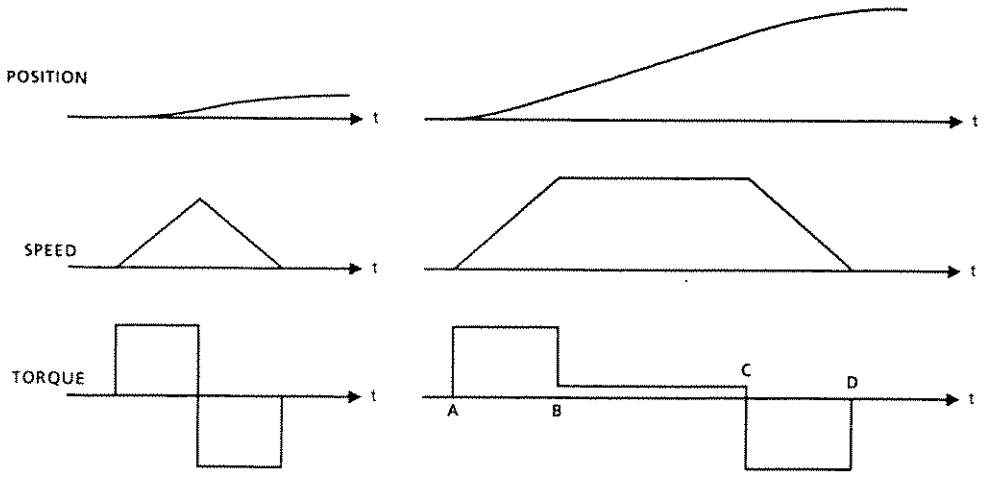

Fig. 107
Position, speed and torque for a setpoint change

You will notice that, at all times during these movements, either the torque/acceleration or else the speed are at the maximum values we can allow, as determined by the design constraints we looked at in the last Item.

For the long position movement, the drive accelerates up to top speed between A and B. Between B and C, the drive runs at a constant speed (top speed), and the torque is much less, only having to overcome running loads (e.g. friction). Between C and D, the drive decelerates back to zero speed and, if this is to be done rapidly, the torque must reverse (i.e. it becomes a braking torque).

DESIGN OF A REAL POSITION CONTROL

The chief feature of real position controls which makes them different from most other types of closed loop controls is that responses for large position errors can be quite different from responses for small position errors. If you just design a position control as a simple proportional control, using the techniques given in Section 1, you can produce a position control which is stable and comes cleanly to position for small movements, but which gives large overshoots (or is even unstable) for large movements.

There are two basic techniques for getting round the problem:

1. Use a ramp on the position setpoint, shaped so that you never ask the drive to move faster, or at a higher acceleration than it is capable of.

2. Use a ramp on the speed setpoint (i.e. on the output from the position control), with position control degaining for large positions errors. You can get a position control designed this way to be stable, with no overshoot in its response, for any size of position error.

In this Item, we consider systems where we feed a speed setpoint to a motor speed controller. For such systems, we will refer to these two techniques as giving 'Type 1' and 'Type 2' position control systems.

PROPORTIONAL CONTROL

Most types of speed control systems (whether in a motor speed controller or in your program) have integral action within the speed control. With such systems, the drive is bound to move when the speed setpoint, calculated by your GEM 80 program from the position error, is at its minimum possible value of one bit. Consequently, the position control will tend to come to a steady state condition with zero position error, even though you have put no integral action into the position controller, but only proportional action.

Consequently, omitting integral action from a position controller does not normally result in any loss of steady state accuracy, and at the same time gets round a number of possible response problems. This applies to either of the two design techniques listed above, which we will now look at in more detail.

TYPE 1 POSITION CONTROLS - POSITION SETPOINT RAMP

This is the easier of the two techniques to program. If you use a ramp on the position setpoint, then what you hope will happen is that the position of the drive will closely follow the position setpoint. As long as it does this, then the position error should remain small, and you will have no peculiar overshoot or response problems.

This technique can fall down on applications where the drive is subject to sudden load disturbances. These could cause the position error to transiently become large, and the sort of response you might then get could seriously overshoot or be oscillatory. If you expect that large transient disturbances could occur on your application, then you should use the second design technique (speed setpoint ramp + degain for large errors) which we deal with later, starting on Page 171.

Assuming that a position setpoint ramp will be O.K. in your application, then you can use the RAMP Special Function S33 on the setpoint SETPOS before you subtract the measured value ACTPOS from it. We will assume that you start off with a purely proportional control. Your control rung can therefore be as below:

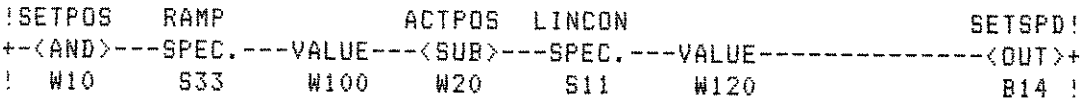

Fig. 108
Program rung with position setpoint ramp

The RAMP Special Function has table values you can set to determine the slope of the ramp, start-of-ramp rounding, and end-of-ramp rounding. By setting these values, you can adjust the top speed, acceleration and deceleration rates of the drive. (Note that RAMP has two separate table values, for the ramp slope when moving away from zero output, and for the ramp slope when moving towards zero output. You should normally set these table locations to contain identical values).

The question now is how to set the gain of the LINCON. First, we note that on a long movement (as in the second diagram in Fig. 107), we must get a speed setpoint output from the rung in Fig. 108 calling for top speed. To get this output with only a small position error, we want the LINCON to have a high gain. (As we said at the start of describing this technique of using a position setptoint ramp, it only works satisfactorily if we can keep the position error small under all conditions).

However, if we make the gain too high, we will also get problems. For instance, suppose we made the gain such that a 1-bit error gave top speed setpoint out of the rung. Then we would never be able to get any intermediate values of output - it would either be

calling for top speed or for zero, and would never be able to call for any speed intermediate between these two. Then, in spite of the S-shaping on the RAMP function on the position setpoint, we could have the speed setpoint alternating on successive program cycles between asking the drive to go flat out and asking it to stop.

Now, you should already know from considering acceleration/deceleration constraints (see last Item) what the maximum step change is which you can allow on the speed setpoint. This will then determine the maximum gain you can have.

Example

Suppose you have worked out that your drive must not decelerate at a quicker rate than twice top speed/second.

Suppose that your program repetition interval is 20 ms.

Then:
Minimum time to decelerate = 1/2 second.
Number of program executions per second = 50
Minimum number of program executions during deceleration = 25
Speed setpoint therefore reduces by not more than 1/25th of top speed per program execution.

That is to say:
Maximum allowable step change in speed setpoint = 4%

Therefore you must not use a gain higher than would give 4% speed setpoint change for a 1-bit position error if you don't want to exceed the given deceleration rate.

Note: Stability might limit you to a lower gain than this. All we have considered above is the maximum gain which you can use without exceeding your deceleration rate figure.

Note in this example that, if you used this maximum gain, then you could expect that, on a long position movement, the position error would reach a value of 25 while the drive was running at top speed. The actual position will be lagging behind the position setpoint as it ramps by this amount, which is generally referred to as the "following error".

Reducing the Following Error

First, let's look at the possible use of a PI control, and see what effect the addition of an integral term would have, because we know that integrals tend to reduce errors to zero (even if they take their time over it). However, we warned on Page 165 that an integral term could give response problems. Fig. 109 shows what the response might be with and without an integral.

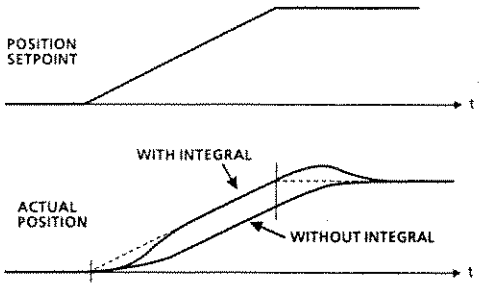

Fig. 109
Position control including an integral term

All that adding an integral term has achieved is to get the actual position to catch up with the position setpoint ramp during long position movements. It has made virtually no difference for short movements. Further, it has introduced two adverse effects.

First, the drive is likely to overshoot at the end of a long position movement. Second, while the actual position is catching up with the setpoint ramp, its slope becomes greater than that of the ramp. As a result, the drive will tend to exceed its design top speed, and probably the design acceleration rate as well. Consequently, an integral term in a position loop using an S-shaped position setpoint ramp is not a good idea.

One method of reducing the lag between the actual position and the position setpoint is to use a feedforward of ramp rate-of-change. As we noted in Item 18 on "Feedforward" in the previous Section, the RAMP Special Function provides a rate-of-change output in one of its table locations. Now, rate of change of position setpoint is (with the appropriate gain) equal to the desired speed at which the drive should be running. If we add this signal in, then all the position error has to do is to provide a position correction, as shown in the block diagram below.

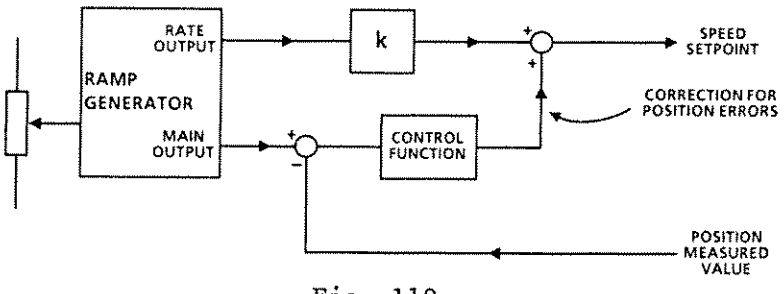

Fig. 110
Position setpoint ramp with rate feedforward

Converting this into a GEM 80 program rung, we get what is shown in Fig. 111.

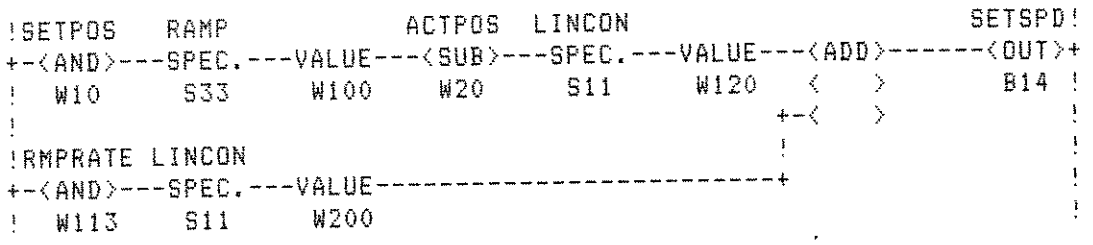

Fig. 111
Program rung with setpoint ramp and rate feedforward

If you refer to the RAMP Special Function S33 software data sheet, you will find that the ramp rate is obtained from table location Zm+13. Thus, if the first table location used by the RAMP is W100 as above, then W113 will be the table location containing the ramp rate. Note that W113 is only recalculated each time that the RAMP function is encountered in the rung. This means that the order of the branches in the above rung is important. If you had written the rung with the branches swapped over as in Fig. 112, then W113 would give you, not what the rate is currently, but what the rate was one program execution ago.

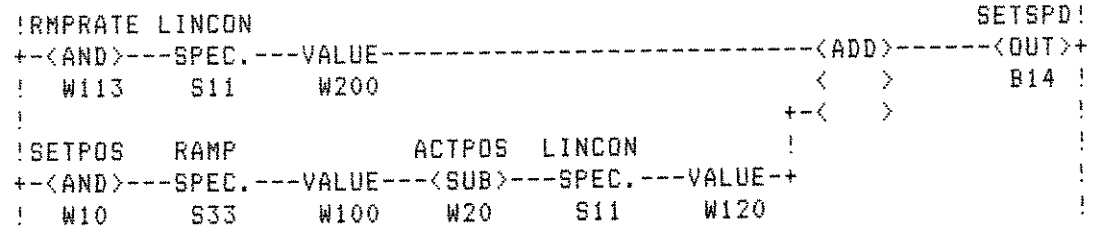

Fig. 112
Incorrect rung with setpoint ramp and rate feedforward

(Note that the contents of W113 in either of the above rungs is the size of the change in the RAMP output, multiplied by 50. Again, see the software data sheet for RAMP.)

In Fig. 111, the gain you put in for the LINCON which acts on the ramp rate can be determined by setting the LINCON following the RAMP to zero gain, and checking that you get the correct top speed on a setpoint change. When checking, you must make the setpoint change large enough for the RAMP to reach its full ramp rate in between the S-shaping at either end of the ramp.

Acceleration/Deceleration Control

When we use a ramp on the position setpoint, the slope of the ramp determines the drive speed. Therefore the rate-of-change of ramp slope determines the rate-of-change of speed, or in other words the acceleration/deceleration. At the start of the ramp when the drive must accelerate, and at the end of the ramp when the drive must decelerate, we therefore need the ramp to be curved, giving

what we normally refer to as an S-shaped ramp. The ideal curve at either end of the ramp is a parabola, since this gives a constant rate-of-change of slope. See Fig. 113.

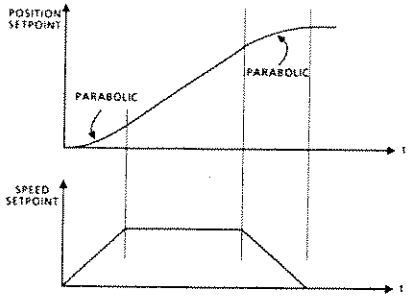

Fig. 113
S-shaped position setpoint ramp

In practice, there are problems in getting a ramp shaped like this, because the standard RAMP Special Function is a general purpose function designed to accommodate changing inputs, and does not provide parabolic rounding at the end of the ramp. If you have a position control where the acceleration and deceleration times are significantly long, you might do better looking at a 'Type 2' position control system.

Protection

So far, we have shown our GEM 80 program rungs directly providing a speed setpoint to the motor controller without any checks on its size. If the position control is performing as we expect, then the speed will be determined by the slope of the position setpoint ramp.

However, under abnormal conditions, we could get the mechanical system sticking or maybe completely jamming. In this case, we need to make sure that the speed setpoint we feed to the motor speed controller is not excessive. We should therefore include a LIMIT Special Function as the last item in the rung before we output the speed setpoint. If we have used rate feedforward as in Fig. 111, then our final version of rung becomes as shown below:

```
!SETPOS   RAMP          ACTPOS  LINCON                    LIMIT          SETSPD!
+-(AND)---SPEC.---VALUE---(SUB)---SPEC.---VALUE---(ADD)---SPEC.---VALUE---(OUT)+
!  W10     S33     W100    W20     S11     W120   (   )    S32     W130    B14 !
!                                               +-(   )                       !
!RMPRATE LINCON                                 !                             !
+-(AND)---SPEC.---VALUE-------------------------+                             !
!  W113    S11     W200                                                       !
```

Fig. 114
Limits added to the speed setpoint

Note that, as we are not using an integral term, we can't encounter any problems from integral wind-up from adding these limits.

(If we did have an integral term, then we would have to limit it separately).

One feature of the LIMIT Special Function is that, while its output is limited, it sets flag bits in the fault word location Zm (W130 in Fig. 114). We can therefore use these flag bits in our program to take appropriate action when the speed setpoint enters limit, such as tripping out the motor controller.

TYPE 2 POSITION CONTROLS - SPEED SETPOINT RAMP AND POSITION CONTROL DEGAINING

We now look at the second position control design technique, where we use a speed setpoint ramp, and degain the position control for large position errors.

Although this technique is a little bit more complicated to program, it has certain advantages over the position setpoint ramp technique we have just looked at:

- The position error can be of any size. If, for instance, the drive sticks a bit, causing the position error to become larger than expected, the response when coming to final position can still be dead beat, which it might not be for a 'Type 1' position control.

- You can use S-shaping on the speed setpoint ramp to limit the rate of change of acceleration/deceleration. This will limit the rate of change of torque, and can therefore reduce mechanical shocks.

If we take a position control using a speed setpoint ramp, then our initial attempt at the design could be as in Fig. 114.

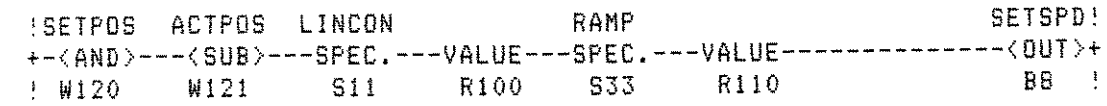

Fig. 115
Possible control rung with ramp generator

The RAMP function in this case (unlike the RAMP in Fig. 108) is limiting the rate-of-change of speed setpoint, not the top speed; that is, its slope provides the acceleration/deceleration constraint. If we try running the system with this rung, we can set up the gain of the LINCON to give quite good responses for small changes of the position setpoint, but we find that, if we give the system large changes of setpoint, we get overshoot. If, on the other hand, we set the gain of the LINCON down to prevent overshoot on long position movements, we get a very sluggish response on small movements.

We therefore need to consider both cases separately.

Small Signal Response

As long as you make small changes of position setpoint only, the normal stability rules of control systems design apply. If you have all the details of the speed controller response, you can analyse the system by any of the standard methods (e.g. Nyquist or Bode diagrams). However, since we normally use just proportional control, then you can set up the control simply by winding the proportional gain up until the drive starts to overshoot when coming to position, and then screwing it back a touch.

Large Signal Response

Let us consider what happens to position, speed, and torque, when the position setpoint is changed by a large amount, as in Fig. 116.

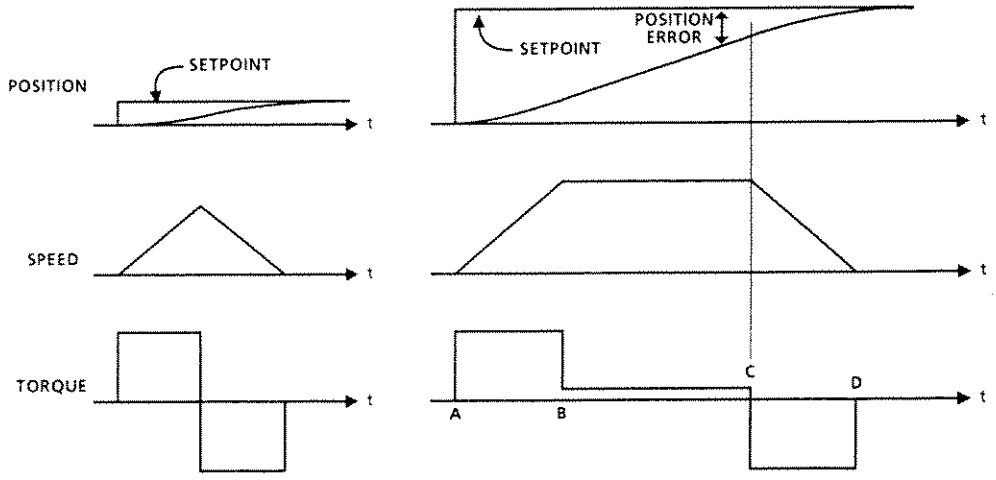

Fig. 116
Position, speed and torque for a setpoint change

This Fig. is exactly the same as Fig. 107, except that in this case we have shown the step change in position setpoint.

What we are particularly interested in is the performance in the time between C and D. How does the control system 'know' when to start braking ? That is, what determines the point C ? If this point is too late, then, since the braking torque we can get must be limited (because of any of the constraints listed in Item 27), the drive will not stop in time, and it will overshoot the desired position. If, on the other hand, we make point C too early, then the system will not be able to use all of the available braking torque without stopping short; in practice, the drive will come to rest rather sluggishly instead of crisply.

What causes the drive to start slowing down at point C is that, at this point, the speed setpoint starts to fall below top speed. Suppose, to start with, that we just have a proportional gain set by the linear conversion Special Function LINCON, S11. To make sure that the speed setpoint cannot go beyond top speed setting,

we need to set the limit values of the RAMP function to correspond to top speed in table locations Zm+5 and Zm+6 (that is, in R115 and R116 in Fig. 115). Suppose that top speed setpoint is 2000. We therefore set the values +2000 in R115 and -2000 in R116 to limit the possible excursion of the output B14.

Next, suppose that we set the LINCON to give a gain of 50, a value we arrived at as the optimum gain from tests for small changes of setpoint. Then, as long as the position error (the difference between W10 and W11) is 40 or greater, the speed setpoint will be equal to 2000. In other words, the drive only starts to slow down when the position error falls below 40.

Next question: Can the drive stop in time if it starts slowing down when the actual position is 40 less than the position setpoint ? To answer this, we need to know what distance the drive covers (i.e. what change in position feedback occurs) when the speed setpoint is stepped from top speed to zero. We can find this out either by performing a test on the drive or by calculation (see later, Page 177).

If we find that the drive can stop from top speed within a position change of 40, all well and good. If, however, we find that the drive covers a considerably larger distance than 40 when coming to rest from top speed, then we have got to modify our program because our position control will, as set, give overshoots on large changes of position setpoint, even though it performed all right on small changes of setpoint.

The first possible modification we could make would be to drop the gain of the LINCON. Suppose our tests or calculations showed that the actual position changed by 100 between top speed and rest if the speed setpoint were reduced to zero, then we could avoid overshoot by dropping the gain of the LINCON from 50 to 20. For top speed setpoint of 2000, the speed would then start to reduce when the position error dropped below 100. Although this would allow us time to stop on long position movements, it would make the performance on small movements slower and unnecessarily sluggish. A better way is to replace the LINCON with a function generator, Special Function FGEN, S30, as in the Fig. below:

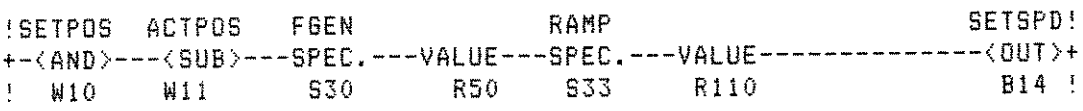

Fig. 117
Control rung with function generator

Note that, in both Fig. 115 and Fig. 117, the RAMP function comes after the LINCON or FGEN. This is because its aim is to limit the rate of change of speed setpoint SETSPD regardless of the LINCON or FGEN settings.

You will notice that, although we began by asking "Can the drive stop in time ?", we have really turned this question into "Can the drive stop in the required distance ?" It is easiest to establish the required function for FGEN by drawing a graph of speed against position error rather than against time.

In this example, we said that the drive needs a position movement of 100 to stop when it has been running with a speed setpoint of 2000. Suppose we reduced the speed setpoint to 1000, and found how far it took to stop if we switched this reduced setpoint to zero. We would find that it was 25. In other words, we have got a square law between speed of the drive and its stopping distance.

(This is, of course, directly derivable from Newton's laws of motion for constant accelerations and decelerations. Since we have a ramp function on the speed setpoint, then our accelerations and decelerations will all be at a constant rate.)

Thus, if we plot the shortest possible stopping distance from any given speed, we obtain a curve which is a parabola, as below:

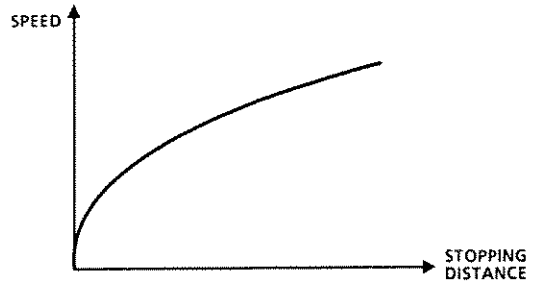

Fig. 118
Curve of stopping distance against speed

To obtain the best stopping performance, we need to set up a function in FGEN which more or less matches this curve. As the drive approaches position, we call for the speed to reduce with position error in such a way that, at any given speed, we know that it can stop within the remaining distance left to go.

Note, however, that the parabola has infinite slope at the origin. We know that there is a maximum gain we can have for stability (in the example above, we said this was 50). Therefore, although most of the function we set in FGEN can be parabolic, we must make sure that the function is just a straight line near the origin, with a slope in this instance no steeper than 50 bits change in output for 1 bit change of input.

Now, let us look in more detail at the function we require, which we will be setting up in tables R51 onwards. First, we need to cater for both directions of travel, and therefore the function must be symmetrical in the first and third quadrants, as shown in Fig. 119.

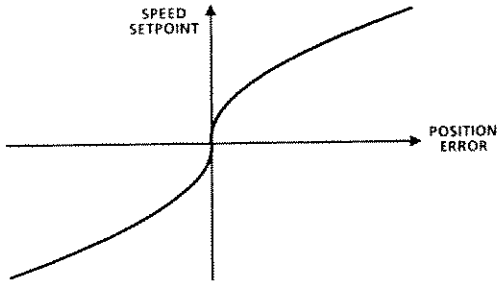

Fig. 119
Position approach from both directions

Second, the way that FGEN works requires that we divide the position error axis into a number of equal increments. Suppose that we make the increment 20, and arrange these increments equally spaced either side of the origin. Then, since we have a square law, we can work out the required values from the equation:

$$\frac{\text{Distance}}{100} = \left(\frac{\text{Speed setpoint}}{2000}\right)^2$$

or: $$\text{Distance} = \frac{1}{40000} \times (\text{Speed setpoint})^2$$

At a position error of 20, the speed setpoint needs to be not more than 894; this will give us a gain for small movements of 894/20, or 44.7, which is slightly less than our LINCON figure of 50, but not much. (Obviously, we could if we wanted to use more segments, with a smaller increment than 20, to allow us to get the full 50 figure). For errors of 40, 60, and 80, the figures of speed setpoint we get from the equation are 1264, 1549, and 1788 respectively, while of course, for an error of 100 the speed setpoint will be 2000.

Note: it is not usually necessary to work the figures out very accurately. If you set them to 1250, 1550 and 1790 you probably would not be able to notice any difference in performance.

These figures have been worked out from the equation of the parabola. All we need to know, to derive this equation, is the stopping distance from any trial speed setpoint.

For any system, the general equation is:

$$\frac{\text{Distance}}{\text{Test distance}} = \left(\frac{\text{Speed setpoint}}{\text{Test speed setpoint}}\right)^2$$

where the test distance is that found by calculation for the given ramp rate (see Page 177) or by tests, for the condition where the speed setpoint is switched from a test value to zero.

The table values for FGEN would then be:

```
R51 :   -100     (= leftmost point)
R52 :     20     (= size of increment)
R53 :     11     (= number of points)
R54 :  -2000
R55 :  -1790
R56 :  -1550
R57 :  -1250
R58 :   -890
R59 :      0
R60 :    890
R61 :   1250
R62 :   1550
R63 :   1790
R64 :   2000
```

Speed Controller Constraint

For many position controls, the only acceleration/deceleration constraint is the torque which the speed controller can provide. If we decelerate in current limit with such a control, we get the required stopping accuracy, the mechanics will stand the torque, and the motor doesn't run into heating problems.

In that case, do we need a RAMP function in our program ? If the RAMP is set to give us acceleration/deceleration rates slower than we can naturally get from running in current limit on the speed controller, then it will be slowing down the response. If, on the other hand, we set it faster than the acceleration/deceleration rates we can get from the speed controller, then the RAMP is not having any effect on the control and is effectively redundant.

However, note that the motor torque is not the only torque in the system. There will also be frictional torques, and maybe load torques. If you let the motor run in current limit during acceleration and deceleration, the motor torque is constant, but these other torques are usually not constant, so you don't get consistent performance.

Friction Torques

If you tried finding the stopping distance of a new drive with no RAMP, and you set up your FGEN on the basis of this stopping distance, then you might find some time later that the drive started to overshoot. This would be because the slides and gears had loosened up as they ran in, and the friction torque had consequently become less. Even where all the mechanics have been run in, you would still probably find that they are stiffer at the beginning of a work shift than at the end, after a few hours running.

As a result, you might find that you had to reshape your FGEN. If, on the other hand, you include a RAMP function, then you can get consistent performance in spite of friction variation. To do

this, you must set the RAMP slope so that it gives a deceleration rate which is not quite fast enough to cause the drive to enter current limit, even when the friction is low.

Load Torques

For most horizontal motions, load torques tend to act as braking torques. For many vertical motions, however, load torques tend to act asymmetrically - they tend to brake upwards movements but give extra acceleration for downwards movements. Unless counterbalanced, hoists, lifts and elevators behave like this. For such controls, the worst case for deceleration is when you come to stop on a downwards movement, where the load is trying to keep the movement going, so that your stopping distance will tend to be longer. On the other hand, for an upwards movement where load assists braking, then the stopping distance will tend to be shorter.

If you use a RAMP which has the same deceleration slopes for the two directions of motion and an FGEN as shown in Fig. 119, then you will have to set these to give correct performance for stopping in the downward direction, for the maximum load torque you are going to apply.

You could improve on the stopping performance in the upward direction, if you wanted to, by using an asymmetric ramp function and an FGEN with different curves in the first and third quadrants. Note that you can't get an asymmetric ramp from the RAMP Special Function without setting its deceleration table value differently depending on direction of motion. Note also that you would have to set the upwards stopping performance for the minimum load.

STOPPING DISTANCE BY CALCULATION

We have assumed up to now that you have discovered the stopping distance of the drive by tests, where you remove the speed setpoint and find how much further the drive moves. However, once you have set the RAMP rate to meet your acceleration/deceleration constraints, it is fairly easy to calculate this distance.

Suppose we have a top speed setpoint of 2000, and that we have set the RAMP so that, on each program cycle, the setpoint reduces by 50. Suppose also that the program cycle time has been set to 10ms. Then it will take 40 program cycles, or 0.4 seconds, for the speed setpoint to fall to zero.

What we need to know next is how fast the position feedback number is changing when the drive is running at top speed (with 2000 speed setpoint). Suppose that top speed is 1500 rev./min, or 25 rev/s., and that our position transducer is a pulse generator giving 100 pulses per revolution. Then, for a speed setpoint of 2000, the position feedback number is changing by 2500 per second. Hence the distance covered, measured in pulse increments, when stopping from top speed is as given overleaf.

distance = average rate of change of distance x time

$$= \frac{2500 \text{ pulses/sec}}{2} \times 0.4$$

$$= 500 \text{ pulses}$$

since, with constant deceleration, the average speed while stopping will be half top speed.

Using the formula given earlier, the required function for our FGEN is determined from the equation:

$$\frac{\text{Distance in pulses}}{500} = \left(\frac{\text{Speed setpoint number}}{2000}\right)^2$$

which is similar in form to the previous example on Page 175.

SUMMARY ON POSITION CONTROL PROVIDING A SPEED SETPOINT

1. As long as your speed controller includes integral action, then avoid using an integral term in your position control. You won't lose out on accuracy as the speed control will make sure that the drive moves for any non-zero position error.

2. The simpler design technique is to put an S-shaped ramp on the position setpoint. The slope of the ramp sets the speed, and the curvature sets the acceleration/deceleration rates. The technique assumes that the drive position can always follow the setpoint with only a small error, It won't necessarily work if the drive can be subject to large load disturbances. Also, the acceleration/deceleration rates will not be accurate unless the ramp rounding is strictly parabolic.

3. The second design technique is to put a ramp on the speed setpoint, and to degain the position loop for large errors. You can usually degain with a function generator set to give a parabolic curve, but with a gain which is not excessive at low errors. The ramp slope sets the acceleration/deceleration rate, which can therefore be more accurate than for the other design technique.

----o0o----

INTRODUCTION

We have so far looked at two techniques for designing position controls. Both of these provided speed setpoints to motor speed controllers. Such controllers normally incorporate their own speed measurement, probably using a tachogenerator, and a closed loop speed control which is often a 2-term PI control. This is a closed loop entirely separate from our GEM 80 program. Further, to prevent excessive currents, most of these speed controllers incorporate current inner loops, i.e. they use cascade control (see Item 17).

Now, it is possible for a GEM 80 system to directly feed the current setpoint rather than the speed setpoint. Effectively, we can move the boundary between the GEM 80 and the drive as below:

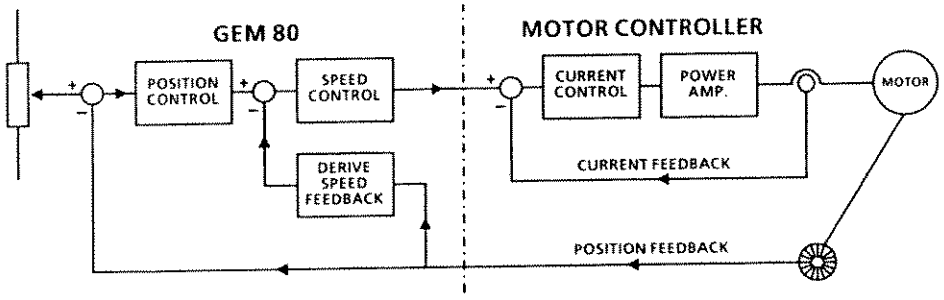

Fig. 120
Providing the speed loop within the program

This arrangement gives certain advantages and certain disadvantages. One disadvantage is that, for drive maintenance, you need the GEM 80 operational. Suppose you try removing the current setpoint signal from the GEM 80 and instead you inject a separate manual setpoint signal. You will find that the drive will accelerate as long as you apply any signal, and you cannot get it to run at a steady speed. So, you need the GEM 80 operational to provide a speed outer loop round the current control.

SPEED MEASUREMENT

With most types of position measuring device, you can get a speed signal as well.

For instance, if you use a pulse generator, you count up the pulses to find how far the drive has moved. If you make sure that the program execution time is regular, then the number of pulses which arrive during any single program execution is proportional to the drive speed. Consequently, you can get a speed measurement without needing a tachogenerator.

Similarly, if you use a shaft encoder which directly tells you the drive position as a number, you can subtract the number on the previous scan from the number on the current scan; this difference will again be proportional to speed (see Fig. 121). So, this is one advantage of getting the GEM 80 to provide the current setpoint - we can eliminate the tachogenerator.

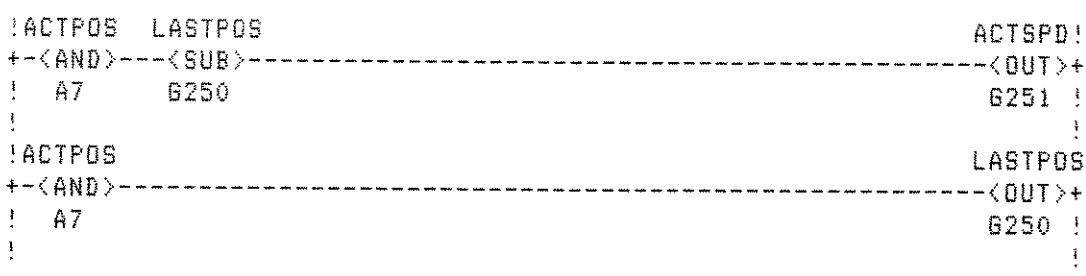

Fig. 121
Derivation of speed signal from position

(The above section of program is really just taking the derivative of the actual position. Compare with Fig. 45, Item 12).

Whichever type of position measurement we use, however, the speed signal we get is not as good as a tachogenerator signal, particularly at low speeds.

For instance, suppose we have a pulse generator which gives us 30000 pulses for full travel of the position control, which takes 10 seconds. Then our pulses are arriving at roughly 3 kHz (this is assuming that the drive spends most of its time on a long movement at top speed). For a GEM 80 program with a 20 ms cycle time, we then count 60 pulses in one program execution.

When, however, the drive runs at 1% of full speed, we only get 0.6 pulses per program execution. In other words, we get a pulse in 3 out of 5 program cycles, and no pulses at all in the other 2 out of 5 program cycles. The GEM 80 program therefore thinks that the drive is starting and stopping rather than running continuously at a low speed. It will therefore tend to increase and decrease the current setpoint rapidly in response to the misleading speed error signal.

Fortunately, a position control of the type we have been looking at, where the drive has to accelerate rapidly up to speed, run at top speed and then rapidly decelerate, will not spend much of its time running at low speeds. Consequently, the poorness of the speed measurement at low speeds does not present any real problem for this type of application.

However, for a position control on a multi-axis system, you might find that you needed to run the X-axis fast and, at the same time, run the Y-axis slow (or vice versa). For such position control applications, you may be much better off providing a speed setpoint to a motor speed controller that uses an analog tachogenerator, rather than attempting to feed a motor current controller.

TYPE 3 POSITION CONTROLS - POSITION SETPOINT RAMP

This is like the 'Type 1' position control we discussed in the last Item. However, instead of feeding out a speed setpoint to

the drive controller, we need to subtract the speed measured value to produce a speed error. We then feed this through a suitable function to generate the current setpoint. To prevent getting excessive current, we also need to limit the current setpoint to what is safe for the power electronics and the motor.

First, we note that the speed loop we are producing includes an integral term in "the plant". As long as we output a current setpoint, the drive speed will tend to increase indefinitely (ignoring friction and load torques). For our control function, we can therefore begin by putting in just a proportional term. We can adjust the gain of this upwards until we start to get the current overshooting on a step change of speed setpoint, when we stop.

However, although we can get a good response with just a proportional term, we would then get a steady state error if load torques or friction torques were applied. To make sure that the speed feedback signal is reset to exactly equal the speed setpoint, we therefore need an integral term as well. Our control therefore becomes a 2-term PI control, for which we can use a PIDABS Special Function. Usually, you won't get any significant inprovement in response by including a derivative term, and you can omit it.

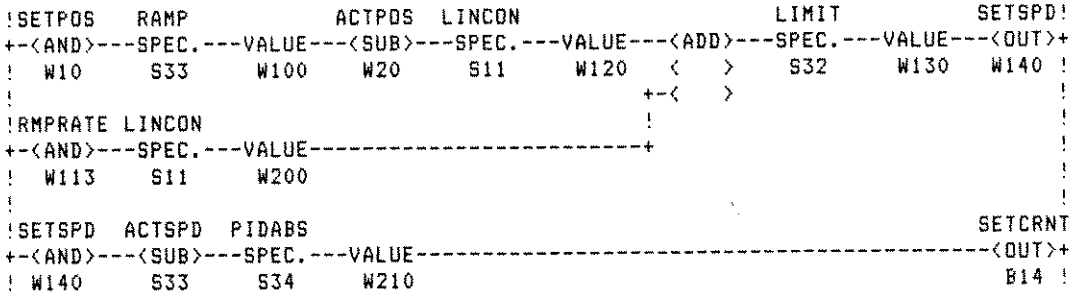

Fig. 122
Current setpoint output rung added to Fig. 114

What we have done above is to add a rear end onto the rung we drew in Fig. 114. We just take the SETSPD output of Fig. 114 and, instead of feeding it out from an analog output module to the motor speed controller, we feed it through another rung first and only then provide an output to a motor current controller.

Now, is there any difference in performance ? The answer is that the difference is hardly noticeable.

As a drive approaches the intended position, the position measurement signal eventually becomes equal to the position setpoint. We then have zero position error, and consequently zero speed setpoint.

However, at the instant when the position error becomes zero, we still have a speed measurement signal, whether we use a tachogenerator or whether we derive the speed signal from the position measurement. In both cases, we therefore continue to get a speed error, and this will be of the correct polarity to call for regenerative current, i.e. the current setpoint will be demanding braking torque. If we have set up the gain of the position control correctly so that we don't get any overshoot, then the drive should stop before the position measurement can change by another bit.

For a system with a tachogenerator into a motor speed controller, the speed measurement signal will slowly decay as the drive comes to rest. For a system where we derive the speed measurement from the position measurement, then the speed signal will last for just one more program execution. As long as the position measurement does not subsequently change, then, since there is no difference between the last position and the present position, the derived speed signal will thereafter remain zero.

TYPE 4 POSITION CONTROLS - SPEED SETPOINT RAMP AND POSITION CONTROL DEGAINING

In the same way that we can turn a 'Type 1' position control into one providing a current setpoint to a motor current controller, so also we can turn a 'Type 2' position control into one providing a current setpoint. We can add exactly the same rung to our program to do this as the second rung in Fig. 122 above.

As for a 'Type 3' position control, it will not work satisfactorily for continous low speed running but, since we expect the position setpoint to move in step changes, we would not normally get this situation anyhow (we would have to drop the speed setpoint limits to do this). Compared with a 'Type 3', we again have the advantage that we always get good long movement responses even if the position error becomes large, and that we can control the rate-of-change of torque by the curvature of the speed setpoint ramp.

SUMMARY ON POSITION CONTROLS PROVIDING CURRENT SETPOINTS

1. You can eliminate the tachogenerator by deriving a speed measured value from the position transducer. If you do this within your GEM 80 program, then you can make your output feed the current setpoint to a motor current controller.

2. However, the speed signal you get is not satisfactory for continuous slow running. Consequently, you should only use a position control providing a current setpoint where all movements accelerate and decelerate rapidly, and spend no significant time running at low speed. Otherwise, you will have to use a position control providing a speed (not current) setpoint, to a motor speed controller with its own tachogenerator as described in the previous Item.

3. You can use either of the two design techniques previously deescribed which provide a speed setpoint. The relative merits of the two techniques were described in the previous Item. You then add on a rung where you subtract the derived speed measured value, and feed this through a PI-control to provide the current setpoint.

----o0o----

INTRODUCTION

If you have a position control carrying human beings, then you need to limit not just the acceleration/deceleration, but the rate-of-change of acceleration/deceleration. If you have some form of carriage moving horizontally, any change from zero acceleration to a finite acceleration as a step change will tend to throw you off your feet. For a vertical motion, such a step change in acceleration causes you to feel that you have lost your stomach.

POSITION SETPOINT RAMP TYPES OF SYSTEM (TYPE 1 AND TYPE 3)

It is virtually impossible to control rate of change of acceleration/deceleration with the ramp itself. To provide this control, we would need some very precise shaping at the start and end of the ramp, as shown below:

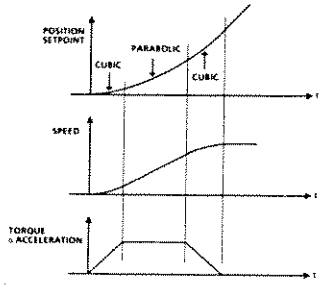

Fig. 123
Curvature in ramp function to limit rate of change of acceleration/deceleration

Since in practice we do not have a true ramp with a digital system but a staircase of setpoint numbers, we need a very high resolution on the setpoint to produce practically smooth parabolas and cubic curves.

When we looked at a simple example (Page 167) where we limited deceleration to a given value, we had a case where a 1-bit position error gave a 4% change in speed setpoint. Obviously, we cannot control rate of change of acceleration/deceleration in a system with such a coarse speed setpoint resolution. We don't want improve the resolution by dropping the gain because this would give us a larger following error. Also, we don't want to use a much higher resolution position measuring device (because of cost, and maybe added complexity in the program if we have to go to more than 16 bits for the position setpoints and actual value).

Therefore we might have to use a second RAMP function, this time on the speed setpoint, with rounding on it at the start and end of any ramp, to provide a limit on rate-of-change of acceleration. This makes the setting up of the overall system more complicated. In particular, we must watch out that, because of the second RAMP function, we don't get the position setpoint from the first RAMP changing too fast at its start and end, otherwise the position error may get rather large and we may introduce overshoot.

SPEED SETPOINT RAMP AND POSITION CONTROL DEGAINING (TYPE 2 AND TYPE 4 POSITION CONTROLS)

With this type of position control system, we already have a RAMP Special Function in the correct place, and don't need to add another one. We just need to set up its rounding table values to limit the rate of change of acceleration/deceleration.

MECHANICAL WIND UP

Humans are not the only things that don't like the sudden application of an acceleration or deceleration.

Most mechanical systems have a noticeable lack of stiffness. The actual stiffness they have is based on engineering compromise. It is obviously more expensive to use thicker shafts, heavier gearboxes, and so on. Further, because these would have more inertia, you might then need a more powerful motor to accelerate them at the rate you require. Therefore mechanical engineers tend to design their mechanical systems to be adequate for the given application rather than an overkill.

Suppose you have a system with a long shaft, with your motor at one end and the load at the other. Then, if you apply the motor torque suddenly, the motor accelerates initially, leaving the load behind. As a result, the shaft twists. (The amount of twist may be imperceptible by eye, but with suitable instrumentation you could measure it). The torque being applied to the load obviously increases as the shaft gets wound up, and the load starts to move. However, once you have finished accelerating, the shaft will then unwind. All this may occur in a fraction of a second, depending on the size of the system. What happens at the end of acceleration is that the speed of the load becomes a decaying oscillation, as shown in Fig. 124.

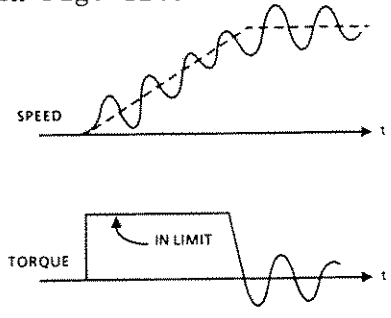

Fig. 124
Mechanical resonance oscillation

There is obviously no hard and fast rule as to what the mechanical resonant frequency will be. For many systems, it tends to be around 15 to 20 Hz.

It is extremely difficult to damp out this type of effect, either mechanically or electrically. Mechanically, you might be able to damp it out a bit using flexible couplings. Electrically, you can

only damp the resonance if you have measuring transducers on both the motor and the load, and even then you may have problems designing a suitable control system. Therefore the best way is normally to avoid giving the system 'kicks' which induce the resonance. That is, you should apply your acceleration and deceleration gradually, using an S-shaped ramp on the speed setpoint, as we described above.

In a similar fashion, you can get resonance effects on lifts, hoists and elevators where the load consists of a cage at the end of a steel rope. The sudden application of acceleration or deceleration can cause the cage to bounce on the end of the rope. The effect can be more difficult to deal with than shaft resonance, since the resonant frequency varies with the length of the rope. Unlike a shaft system, resonance is therefore not at a fixed frequency. Again, the best approach is normally to avoid giving the system any kicks of acceleration or deceleration.

SUMMARY ON AVOIDING MECHANICAL SHOCKS

1. On some types of position controls, you need to avoid the sudden application of torques, as these give step changes of acceleration and deceleration. Particular cases where you need to avoid step changes are:
   - People-moving systems.
   - Systems with springy mechanics.

2. For a 'Type 1' or 'Type 3' system using a position setpoint ramp, it is virtually impossible to get the curvature you require on this ramp, unless you use a very high resolution on both the setpoint and the position measurement. Even then, the combination of parabolas and cubic curves you need is difficult to program.

   Instead, it is better to use an additional speed setpoint ramp with rounding to limit the rate of change of acceleration/deceleration. Care must be taken on setting up with the two ramps, so that the position setpoint ramp doesn't beat the actual position.

3. For a 'Type 2' or 'Type 4' system using a speed setpoint ramp and degaining for large position errors, you don't need to add anything extra to the system. You just set the ramp rounding to limit the rate of change of acceleration/deceleration.

----o0o----

IN-POSITION DETECTION

To detect when a drive has reached position, you can check when the position error is equal to zero. However, if you want to initiate some other action when the drive has reached position, this may be unnecessarily late. You may instead be able to check when the position error lies within a narrow band. This will give an indication, as the drive approaches position, slightly before it comes to rest, and thus cater for time delays in operating solenoids, etc.

BRAKES

For some types of mechanism, holding brakes are used to prevent any movement of the mechanism after it has run to position. This is a particular example of where, to avoid delays, initiation of dropping on the brakes should usually occur before the mechanism has reached final position. If, however, too much anticipation is used, then the brakes will drop on early and assist in stopping the drive. This might cause it to stop short, and would increase brake wear.

If the drive is subject to an overhauling load, you may well need brakes to hold the drive when it reaches position. Otherwise, the motor would have to remain stationary producing a torque to back off the load. Not all motors are rated for being stalled in this way as, when not rotating, cooling may not be adequate.

For such applications, you must also always look at how to lift the brakes as well as at how to drop them on. If you lift the brakes immediately the position setpoint is changed, the load torque might start to cause rotation before the motor has time to develop a corresponding electrical torque, and the drive may start off rotating backwards, for a brief moment. This is more likely where you have got a RAMP function limiting how fast the speed setpoint can rise. You may therefore need to delay lifting the brakes (perhaps by only a fraction of a second) on changes of position setpoint.

However, you can also get the reverse effect if you lift the brakes too late. In this case, the motor will already be developing a torque bigger than the load when the brakes come off, and the drive will snatch, giving an initial high acceleration kick. Since many brakes have fairly high inductance coils, and the coil circuits therefore can have long time constants, you may need to initiate the brakes ahead of the position setpoint being changed.

DRIVE INHIBIT

Most commercially available types of motor drive control system use predominantly analog circuitry, with an analog speed measurement signal from a tachogenerator, and requiring an analog setpoint signal as input. Apart from the cost, this is largely because, as we saw in Item 29, there are no simple ways of

providing a digital speed measurement signal with a fast respose on speed changes down to zero speed. As most motor speed controllers are designed for general use (not just for position controls), then they need to be able to operate at continuous low speeds, and so need to use analog tachogenerators.

A problem with analog circuits is that they drift. If you put zero speed setpoint into a motor speed controller, you may find that after several seconds the motor gives a kick, rotating by a fraction of a revolution, after which it remains stationary, and then the process repeats. Some types of controller have a zero offset adjustment pot, and adjustment of this can increase the time between movements. However, if you don't want any kicks at all, you may need to inhibit the controller. Most controllers have an input for this purpose, which usually has to be energized to enable the drive to run.

In your GEM 80 system, you should therefore include an output to the controller inhibit, and only enable the drive while it is moving, or while it is stationary for short periods, but you should inhibit the controller again if the drive is going to remain stationary for more than a few seconds. Don't, however, put on the inhibit until the motor has come to rest or you will lose out on braking torque too early, and the drive may overshoot.

If your system includes holding brakes, don't drive the inhibits directly in parallel with the brakes. You will usually need to delay putting an inhibit on till after the brakes have actually come on. Depending on how long it takes brakes to lift, you may need to lift the inhibit before or after energizing the brakes.

Remember that, when a drive is inhibited, it cannot produce any torque. If you have no holding brakes, then any external torques applied to the drive while it is inhibited may drive back and cause rotation. In such cases, it may not be possible to inhibit the motor controller, and the motor will have to provide electrical torque at standstill to back off the external torque. Check that the motor is rated for such a duty, as if it is not then holding brakes will need adding to the mechanical system.

DATUM SETTING

Where you are using a position transducer which is incremental (e.g. pulse generator) or absolute with turns counting, then datum setting is necessary on power up.

However, it is generally to be recommended that you design your system so that the datum is reset on a regular basis, and not just on power up. This guards against the datum getting lost from any hazards, such as electrical noise interference creating spurious pulses which are large enough to get counted.

For instance, many position controlled machines perform a sequence of movements and then run back to a "home" position. For such

machines, datum resetting can usually be arranged to take place on the run back to this home position.

----o0o----

INTRODUCTION

Once upon a time it was common practice to use a big motor and lots of layshafts and bevel gearboxes. The gearboxes kept everything in step and therefore synchronized.

However, shafts and gearboxes require maintenance. Nowadays, it is therefore common practice to use separate small motors and to synchronize them electrically. We have two cases to consider:

- Where two drives are both position controlled, following the same position setpoint.
- Where the positions of two drives do not matter, as long as they keep in step with each other.

COMMON POSITION SETPOINT

For a system where two drives must both follow the same position setpoint, the simplest system is to use a 'Type 1' or 'Type 3' position control. We use a common position setpoint ramp, but we have two separate position measurements (ACTPOS1 and ACTPOS2). For a 'Type 1' system outputting speed setpoint to each of two motor speed controllers (SETSPD1 and SETSPD2), and using a feedforward from ramp rate to minimize the following errors, we get the following program:

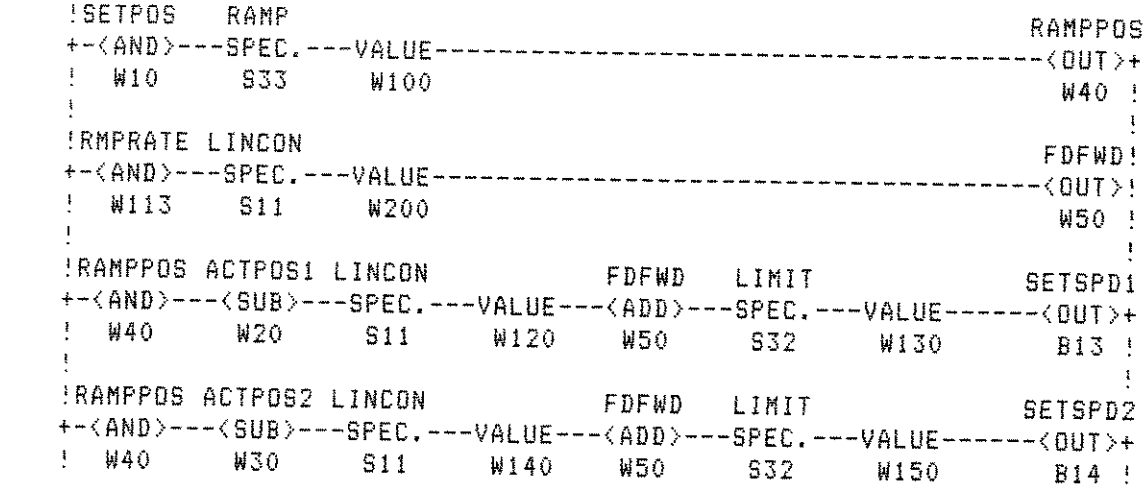

Fig. 125
Position control giving speed setpoints for two drives

As we mentioned on Page 171, we can make use of the flag bits in the fault words provided by the LIMIT Special Functions to take action if the position error on either drive becomes excessive for any reason.

For a system using a 'Type 2' or 'Type 4' position control (which we might have chosen as being easier to program to avoid mechanical shock), we cannot synchronize two drives so easily. Instead

of the two drives following a ramping setpoint as in Fig. 125, they both receive (maybe big) step changes in setpoint. The way in which they reach the desired position is then left to their own devices. To synchronize them, we need a separate control loop which acts on the error between the actual positions of the two drives, and provides a speed setpoint correction to one or other of them to pull them into step. Now, this is really the same type of system as our second case of keeping two drives in step where their absolute position doesn't matter, which we deal with below.

SYNCHRONIZING TWO SEPARATE DRIVES

Let's begin with the case of two drives in a position control 'Type 2' or 'Type 4' receiving a common position setpoint. We can easily find the relative error in position by subtracting the one position measurement from the other, as below:

Fig. 126
Forming the relative position error

Before we consider how we are going to stabilize our control, we need to decide where its output is going to go.

The first possibility is to feed the output into just one of the drives, adding to the normal speed setpoint, to buck or boost its motor speed. Well, this would work O.K. as long as we weren't running at top speed. If we were, then, on trying to boost the speed, we would tend to get overspeed. If we had prevented this by using a LIMIT Special Function, then we wouldn't get any change in speed at all, and we wouldn't get our two drives to synchronize.

So, a second possibility is to feed our relative position error into both drives. We feed it directly into one drive, but with its polarity reversed into the other. This will buck the speed of one and boost the speed of the other, or vice versa depending on the polarity of the relative position error. Now, if one drive is at top speed and can't be boosted, we can still buck the speed of the other, and pull the two drives into synchronization.

For most applications, this second possiblity is usually good enough. So, how do we stabilize the loop ? First, let's assume that the two motors with their motor speed controllers are identical. If we now run with our control open loop and apply a small speed correction signal, causing one drive to run faster and the other to run slower, then the relative position error will continuously grow. In other words, our system contains an integral term, and is an example of what we called Special Case 5 (see Item 14, Page 77). To get a good response, all we need is a proportional gain in our control.

However, in practice our two drives will not be identical. For instance, we may find that, to get them to run at precisely the same speed, they need fractionally different speed setpoints. If we have our two drives synchronized, then this difference must be provided by our relative position error. Since we want this to be zero, we must therefore include an integral term. That is, we need a 2-term PI-control. (If we only included a proportional term, then the error would not increase, but would remain constant rather than zero for running at a steady speed).

Fig. 127 shows the resulting system, using a PIDABS Special Function to give us the PI-control.

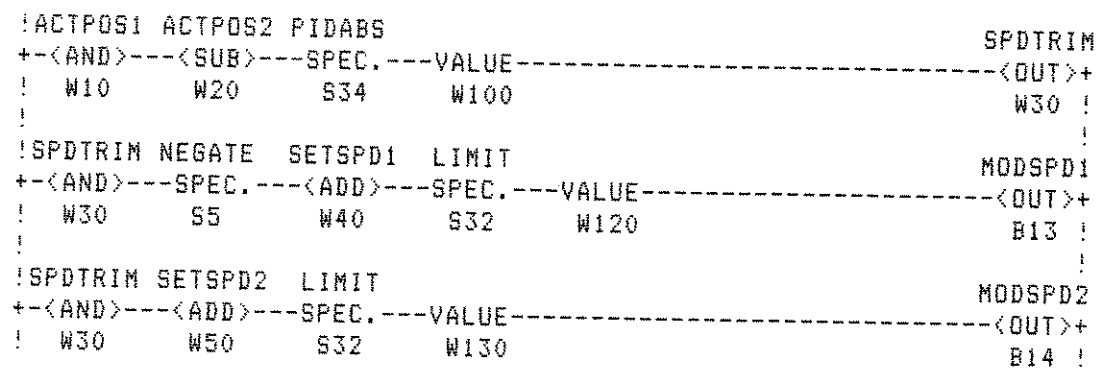

Fig. 127
Position tie loop

In the first rung, we will get a speed correction SPDTRIM which is positive if the position of the first drive ACTPOS1 is beating the position of the second drive ACTPOS2. In this case, we need to increase the speed of Drive 2 and reduce the speed of Drive 1, so that Drive 2 can catch up. We therefore include the NEGATE Special Function in the second rung to reverse the sign of the SPDTRIM speed correction.

On the other hand, if the second drive is ahead, we will have a negative value for SPDTRIM, and the speed of Drive 2 gets decreased while the speed of Drive 1 gets increased.

Note that, with the PIDABS Special Function forming the speed trim, we can limit the maximum size of the speed trim, and we can also use the flag bits in the fault word to give an indication if it tends to become excessive for any reason.

The system of Fig. 127 above is not quite perfect. When the motor is running below top speed, we can buck the speed of one motor and boost the speed of the other. If one drive is at top speed, however, we can only buck the speed of the other, and the rate at which the speed error reduces is therefore only half what it was. We therefore get a 2:1 variation in gain. A better system is therefore one which bucks the faster drive only, and doesn't boost speeds at all. We don't then get any change in gain when one of

the drives is at top speed, so we can set up the system to have a fast response under all conditions.

However, if you now reverse the direction of rotation of your drives, what was a bucking correction becomes a boosting correction, so you have to take the polarity of setpoints into account. We have not illustrated such a system since, in principle, it is the same as Fig. 127, but with some additional switching logic to block any boosting speed trim signals.

CONTINUOUS ROTATION TRANSDUCERS

Now, suppose that the drives are not position controlled. We can use a position-tie loop exactly as we have described above. However, if the drives are capable of continuous rotation, then our position transducers must also be of a type which allows continuous rotation.

Absolute Position Transducers

If we use absolute position transducers (see Item 26), then the numbers they produce will recycle.

For instance, if you have a 12-bit shaft encoder, the number read by the GEM 80 may climb from 0 to 4095, suddenly drop back to 0, and then start again. Next, assume that you have two drives which are one bit out of step in position. When the position transducer on the first drive produces 4095, the position transducer on the second drive produces 4094, and the difference is +1. For a movement of one bit more by both drives, however, the first transducer gives 0 and the second transducer gives 4095, making the difference all of a sudden become -4095. After another movement of one bit, the transducers will give 1 and 0, and the error becomes +1 again.

We therefore have a problem with relative position errors where the numbers recycle, but this is easily overcome by adopting the following strategy:

- If the error is within the band ±half the number span, then we accept the error as it is.
- If the error is greater than +half the number span, we subtract the number span from it.
- If the error is less than (i.e. more negative than) -half the number span, then we add the number span to it.

This allows our transducers to be up to half a revolution out of step. (If the error ever gets beyond half a revolution, the drives will tend to synchronize a whole revolution out of step. We should not, however, get such a situation occurring in practice unless one of the drives jams mechanically).

To verify the above strategy, we tabulate overleaf how the error will progress with two transducers one bit out of step in either direction.

Transducer 1 (leading)	Transducer 2 (lagging)	Error (T_1-T_2)	Corrected error
4094	4093	1	1
4095	4094	1	1
0	4095	-4095	-4095+4096 = 1
1	0	1	1
2	1	1	1

Transducer 1 (lagging)	Transducer 2 (leading)	Error	Corrected error
4093	4094	-1	-1
4094	4095	-1	-1
4095	0	+4095	+4095-4096 = -1
0	1	-1	-1
1	2	-1	-1

Where, however, the number span of the position measurement signals is 16-bits, then we hit a new problem. Suppose we have absolute position transducers that give 16 bits. When we connect these to a GEM 80, the numbers they produce will be interpreted as being in the range -32768 to +32767. So far so good. However, if we have two such transducers, one on each of the two drives we are trying to synchronize, then we could find ourselves wanting to subtract +32767 from -32768. Because the number range used in ladder diagram calculations in a GEM 80 Controller is +32767, the answer we will get from the SUB| calculation is -32767, not -65535. At the same time the overflow flag bit E0.7 is set.

This is no good for working out what the relative position is between the two transducers. We therefore have to limit the signals from the transducers to 15 bits for the position tie system, which means discarding the least significant bit.

One way we can achieve this is by connecting the most significant bit from the transducer to bit 14 (not bit 15) of the input word, connecting all the other wires one bit down (so that we connect the next-to-least significant bit to bit 0), and not connecting the wire from the least significant bit at all. Another way is by connecting the transducer normally and then dividing the input number by 2. Either way, we will finish up with position numbers in our GEM 80 program which are always positive, and therefore in the range 0 to +32767. When we come to subtract one such number from another, then we can never get an answer more negative than -32767, which is representable by 16 bits in the standard way.

Incremental Position Transducers

You could follow exactly the same practice as we have just outlined if, instead of using absolute position transducers, we were using incremental transducers such as pulse generators. For synchronization, however, there is little point in counting pulses from each pulse generator. Instead, to avoid the big-number prob-

lems and recycling numbers problems we got with absolute transducers, we can subtract the numbers of pulses counted in each program execution, and total these incremental errors, as in Fig. 128.

Fig. 128
Forming the relative position error with pulse generators

If the drives are running at the same speed, then the numbers read from counter inputs P/GEN1 and P/GEN2 will be identical on any program execution, so we just put the relative position error RELPOS back in its original table location. (Note: We are assuming that the counter input modules used are set up so that their registers are cleared to zero each time they are read). If, however, the drives are not matched in speed, then one of the pulse generators will give a bigger number than the other, so RELPOS will increase or decrease correspondingly.

To produce a position-tie loop, we can use the same type of system as for Fig. 127, except preceded by the rung of Fig. 128 and with the PIDABS Special Function operating directly on the error.

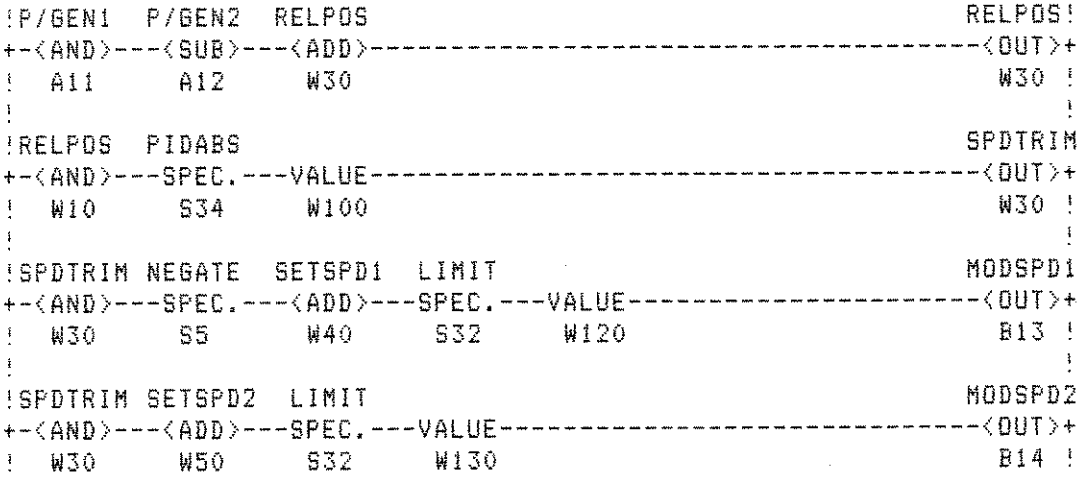

Fig. 129
Position tie loop with pulse generators

As for all incremental position measurements, we need to know where we are at start up. For a position-tie system, this simply means that, when we start running, the position of the two drives relative to each other must be correct. We can then set RELPOS to zero if it isn't already. After that, the two drives should keep synchronized without RELPOS becoming anything other than a fairly small number, and we don't run into big number problems of the type we met with absolute transducers.

The only possible problem we could encounter is loss of pulses. For normal position controls using incremental transducers, we said that if possible you should re-zero from time to time when the drive passed a datum-setting limit switch. However, when you have two drives rotating continuously, a mechanism for resetting the relative position error to zero is not simple. Therefore, if you have an application where loss of synchronization could halt a production run, you probably ought to consider using absolute position transducers.

SUMMARY ON DRIVE SYNCHRONIZATION

1. Position controls which use a common ramped position setpoint naturally tend to remain in step in position, without any additional program.

2. Position controls where the common position setpoint moves stepwise need an extra position-tie loop. This modifies the speed setpoint of one or other drive depending on the error in relative position between the two drives. The control will generally be a PI-control.

3. For drives where their relative position is important but their absolute position is not, you might want the drives to rotate continuously. Your position transducers must then be capable of continuous rotation.

4. With absolute position transducers, their numerical outputs will recycle. You need to be careful about handling the error between two transducers as they run through the recycling point, and the maximum size of numbers used.

5. With incremental position transducers, you can avoid large number problems by subtracting the increments for each program execution, and only totalling these differences.

----o0o----

INTRODUCTION

If your position control is not powered by an electric motor drive, but is driven by a hydraulic ram or hydraulic motor, most of the same principles apply as have been given previously. There are, however, one or two differences which may need some additional programming or additional control equipment.

The way you select your position transducer will be no different whether the power is electric or hydraulic.

DEAD ZONE

Most hydraulic proportioning valves are of the spool type, with a centre off position, and two separate solenoids, one causing the valve to allow fluid flow to one side of the motor or ram, and one causing fluid flow to the other side. In either case, the valve will also provide a return path to the sump. In diagrammatic form, the arrangement will be as below:

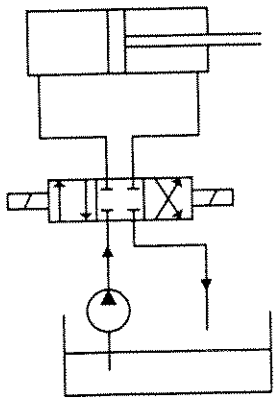

Fig. 130
Hydraulic position control

However, the valve may require a signal corresponding to a small position error before any flow begins. You may therefore need to include the Special Function DEDBAND, S31, into the position control rung. For instance:

```
!SETPOS  ACTPOS  LINCON          DEDBAND                        VALVE!
+-<AND>---<SUB>---SPEC.---VALUE---SPEC.---VALUE-------------<OUT>+
!  W10     W11     S11     W50     S31     W60                 B8  !
```

Fig. 131
Control incorporating a dead band

(As we have seen previously, LINCON might be replaced by FGEN if we were using the equivalent of a 'Type 2' position control). You should set up the characteristics of the DEDBAND Special Function as in Fig. 132.

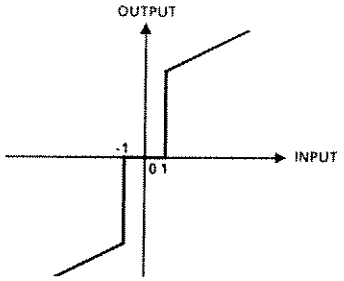

Fig. 132
DEDBAND function settings

Where the stopping distances from top speed are fairly short, it may be possible to program the deadband characteristic into an FGEN together with any other required shaping. To do this, the FGEN increment between points must be 1, and so it would need a lot of data values if the stopping distance from top speed were more than a small figure, e.g. 10. For larger figures, it normally requires less data to set up the required characteristic using separate DEDBAND and FGEN functions.

WATER HAMMER

Because hydraulic fluid is (virtually) incompressible, hydraulic systems do not like valves being opened and, more important, shut rapidly. Such rapid changes can cause water hammer effects. These are more likely in long pipework runs. Normally, damage is prevented by having pressure relief valves, but it is obviously not desirable for these to have to operate under normal working conditions.

To smooth out valve movements, it is therefore preferable to have some form of ramp or smoothing of the signals to the valve solenoids. Even with GEM 80 systems with fast program cycle times, the use of the RAMP Special Function S33 may not be good enough. RAMP will produce an output varying as a staircase, and even staircases having very short steps may not be smooth enough for the hydraulics.

Most manufacturers of proportioning valves also produce suitable driver units which include an analog ramp. These units also will accept a single analog signal and, depending on its polarity, drive the appropriate valve solenoid.

DITHER

Proprietary driver units often include a certain amount of dither on the valve, by feeding a signal at a high frequency alternately to the two valve solenoids. This dither is normally insufficient to linearize the valve operation. It is primarily intended to prevent the valve remaining stationary for any length of time, when it might tend to stick. Consequently, you still normally need to use a DEDBAND.

----oOo----

SECTION 4

MISCELLANEOUS TOPICS

CONTENTS

----o0o----

INTRODUCTION

In previous Sections, we have frequently mentioned taking recordings of the measured value, to find out how it responds to a change in setpoint, for both open loop and closed loop systems. In effect, we have just said "take a recording" without considering the practicalities of how easy or difficult it is to take such a recording.

Since there are other cases where we may want to take recordings besides setting up closed loop controls, we will consider the subject in this Item in a little more detail.

DIRECT ANALOG RECORDING

For many process measurements, we can use an analog recorder connected directly to our measuring transducer or to the output side of any signal conditioning transmitters.

Where it accepts a 20mA signal, a GEM 80 analog input channel on a module makes use of a precision resistor connected as a shunt. This resistor is usually mounted on a termination panel or rail, not on the module itself. That is, the module is really a voltage input module monitoring the voltage drop across the resistor. For recording purposes, you can use this shunt resistor similarly, to provide a voltage input to a recorder. Make sure, however, that your recorder has a high input impedance. The current it draws must make negligible difference to the voltage drop across the resistor, otherwise you may affect the control of the plant. If in doubt, insert a second resistor into the current loop in a suitable place. As this resistor will be independent of the control system, you don't then have to worry about the current drawn by the recorder, except as regards the recorder calibration.

However, we cannot make direct measurements in all cases. If we are linearizing signals in our GEM 80 program, then what we need to record is table values from within the program after linearization, and not the voltage inputs to the analog modules. For pulse train signals, we cannot record pulse frequency as such, only the individual pulses; we could then work out the frequency by measuring how many pulses occurred in a given time, but this is not easy to do from a recording for fast pulse rates.

For most response recordings, we need to know when the setpoint was changed, even though we don't necessarily need to know the size of this change (see Fig. 18, Page 29). Since most setpoints for GEM 80 Controllers come from thumbwheel switches, keypads or keyboards in digital form, then again we cannot record such setpoints directly, but need to get at table values within the GEM 80.

USE OF SPARE ANALOG OUTPUTS

For recording purposes, it is useful if the GEM 80 Controller has a spare couple of analog outputs to which a recorder can be connected. In fact, for some applications, you may feel it worthwhile to have a couple of dedicated analog output channels built into the equipment for recording purposes only.

If all you need to know on a recording is at what time a setpoint was changed, then you could record this off a spare logic output module, and you only need a single analog channel for recording the measured value. However, if you need to record the size of the setpoint as well (e.g. if the plant contains an integral term, our Special Case 5), then you really need two analog channels.

It is then a simple matter to put into your program a couple of rungs to copy the table values of the setpoint and measured value to these analog output channels.

DIGITAL LOGGING

As an alternative to recording analog signals with a pen recorder, you can put in your program some rungs to copy the table values of your setpoint and measured value into a succession of spare workspace table locations. For instance, you could get the program to copy W10 into W1000 on the first program cycle, into W1001 on the second program cycle, into W1002 on the third, and so on. Obviously, you can't go on for ever in this fashion; you will have to include some form of limit to stop the process before you have filled up all the available memory.

One useful method of storing the last few values of a particular table value is to use the LOCATE Special Function S20 and the MOVE Special Function T20, using an overlapping move. For instance, suppose we want to place the last twenty values of W20 in table locations W1000 to W1019. Then we could do this with the following rungs:

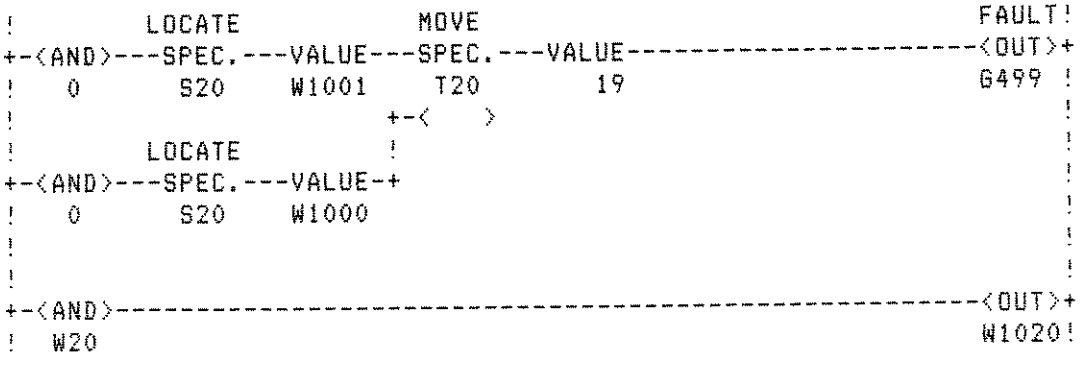

Fig. 133
Rungs for historical logging

The first rung moves the contents of W1001 into W1000, W1002 into W1001, etc., through to W1020 being moved into W1019. The second rung then puts the latest value of W20 into W1020. Thus, the contents of W1000 to W1020 are the values which W20 had from 20 program executions ago up to the value it had on the present program execution.

If you want to record what was happening just prior to a particular event, then you could put the rungs inside a BLOCK. You could then arrange for the BLOCK to be executed under normal conditions on every program cycle, but to stop the BLOCK from being executed further if the particular event occurred. In this way, you would "freeze" the data held in W1000 to W1020. You then have a record of what was happening immediately prior to the particular event.

This type of technique is obviously much more economical in storage space (i.e. the amount of memory required) than trying to record values continuously. Note, however, that if you try logging too many historical values, then the time taken by the MOVE Special Function shunting the values in the table locations could become rather long, and make your program execution time slow.

The simplest way of accessing the data would be to display it on a Programmer screen in datalist display mode. Other methods of accessing it depend on what facilities are provided by your GEM 80 Controller, and what you write in your program. For instance, you could get the values printed out on a printer, or displayed on a video screen (maybe as a graph or bar chart), or, if you had a spare analog output module, you could play it back onto a pen recorder.

----o0o----

INTRODUCTION

For some types of control, you may want to try the control system out without the real plant, but with it simulated by program. You can then try out what the effect will be of changing the plant operating conditions, and how the various process variables behave when there are changes of setpoint or disturbances.

PLANT MODEL

You can only carry out such a control system simulation if you know sufficient about the plant to be able to model it reasonably accurately. You need to know what the various time constants, time delays and gains occurring in the plant will actually be. The more accurate your data about the plant, the more accurately you can model the plant and simulate what will happen in practice.

BASIC TOOLS

Time constants:
Simulate with ANALAG Special Function S37.

Time delays:
Simulate with STORE Special Function T30. (You could also simulate a time delay with the overlapping MOVE, as given in the previous Item on "Recording Techniques". However, the execution time of STORE is shorter, as this Special Function only writes to and reads from one table location per execution).

Gains and offsets:
Simulate with LINCON Special Function S11.

Non-linearities:
Simulate with FGEN Special Function S30.

SCALING OF PROCESS VARIABLES

The first thing to do in a simulation is to decide what numbers in your plant model you are going to use to represent the physical values of the process variables in the real plant. To allow for overshoot in responses, you should not make use of the maximum numbers of ±32767 for maximum values of the process variables, but make them, say, about 20000.

LONG TIME CONSTANTS AND LONG TIME DELAYS

Note that you can make a long time delay by cascading a number of shorter time delays. However, you can't do this with time constants. Each time constant must be represented by a single ANALAG. If you run out of range on the times you need, then you must put the ANALAG in a time-dependent BLOCK, so that it only operates at discrete time intervals, rather than operating on every program execution.

SCALING OF TIME

One trick you can do on a simulation, which you can't do on a real plant, is to speed everything up by a factor of 10, say. If you make all your plant model time constants and time delays 1/10th of those on the real plant, and if you also make your integral and derivative times in your PID controls 1/10th of their calculated values, then you can produce a simulation operating 10 times as fast as the real plant. This can be very convenient where your plant has very long time constants and time delays, and where you might otherwise have to wait several minutes (or hours) to see how the system responded.

Note that, when scaling time, you must leave all the proportional gains in your controllers and plant model as they would be for real time. It is only the time dependent values you must alter.

QUANTIZATION

If you want to simulate the effect of ADC's or DAC's where, for instance, they have 12 bit resolution, then you can divide by 16 and multiply by 16, using the integer arithmetic Special Functions DIV (T11) and MULT(T10), e.g.:

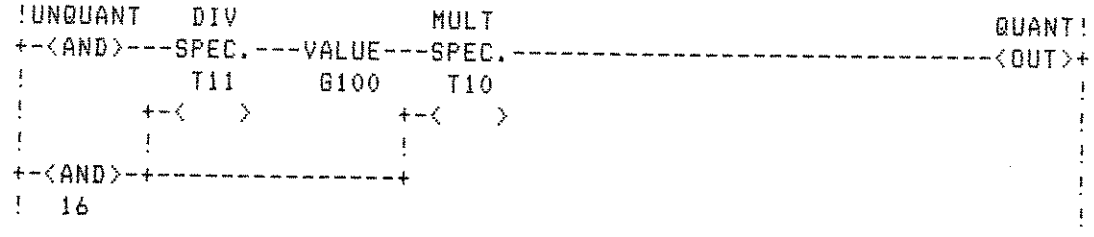

Fig. 134
Simulation of quantization

In the above rung, the remainder of the division in G100 is ignored. Multiplying the DIV output by 16 therefore gives a rung output which can only move in steps of 16, and therefore can only take on values of $\pm$2047 x 16. This gives you the equivalent of a 12-bit DAC or ADC.

----o0o----

PROPORTIONAL BAND

The reciprocal of what we have called "loop gain" for the proportional term in a 2 or 3-term control, expressed as a percentage. Thus, if we have a control where the loop gain provided by the proportional term is 7, then the proportional band is 100/7, or 14%.

Thinking of this another way, we can say that, with a proportional band of 14%, we would need to change the setpoint by 14% of its span to get a measured value change of 100% of its span, if the control loop was opened.

RESET ACTION

Means that the controller contains an integral term. After a change of setpoint or a disturbance, it causes the measured value to be reset exactly equal to the setpoint. A controller without reset action will generally have a slight difference between the setpoint and the measured value under steady state condtions.

RESET TIME

Also called integral time.

For a controller with an integral term, the reset time is the time taken for the integral term to give an output equal to that given by the proportional term.

Thus, if we have a PI-control with a proportional band of 14%, then a 1% change of setpoint will give about 7% output immediately due to the proportional term. The integral term (the reset action) will in addition cause the output to ramp up. The reset time is the time when the output has ramped up a further 7%, as in the Fig. below:

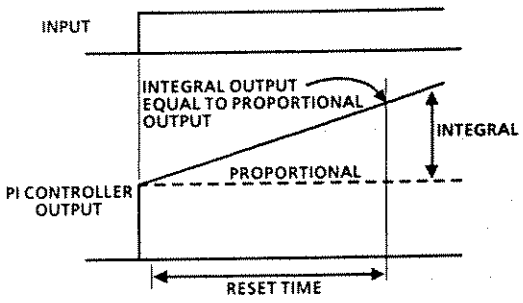

Fig. 135
Reset time

DERIVATIVE ACTION

Means that the controller contains a derivative term. This may operate on the error, or just on the measured value.

DERIVATIVE TIME

This is rather like the reset time worked out back to front. If we have a controller with a derivative term, the derivative time is the time for the input to ramp from zero to a level which is equal to the level giving the same output as the proportional term.

Thus, if we have a PD-control with a proportional band of 14%, we get about 7% output for a 1% input due to the proportional term. To get the same output of 7% from the derivative term, we would need the input to ramp up. The derivative time is the time at which the input becomes 1% when the ramp rate has been correctly adjusted to give the 7% output, as in the Fig. below.

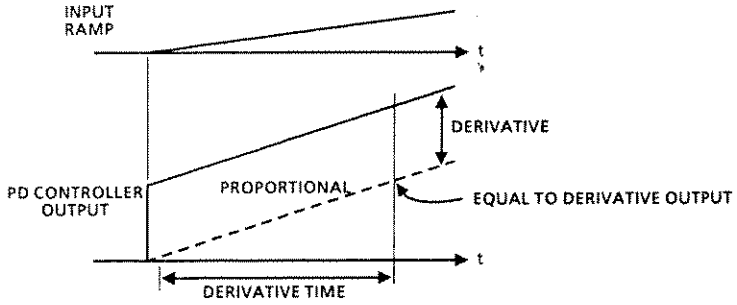

Fig. 136
Derivative time

----o0o----

INDEX

Main topics are shown underlined.
'f' means 'and the following page'.
'ff' means 'and the following pages'.

FEEDBACK

We expect to start working on the next edition of this manual even though we have only just published this one. To make the next edition better, please let us know what shortcomings you found in this edition.

When you're having a cup of coffee, perhaps you could jot down a few points on the page overleaf, and mail to:

Dick Wilson
Head of Systems
Microprocessor Products Department
GEC Industrial Controls Ltd
Kidsgrove
Stoke-on-Trent
ST7 1TW
England

or else pass on to your local GEM 80 agent.

CLOSED LOOP CONTROL SYSTEMS
USING GEM 80 PROGRAMMABLE CONTROLLERS
(1987 Edition)

Feedback from:

Name .....

Position .....

Company .....

Town or city .....

County or state .....

Postcode/ZIP .....

Country .....

Items which ought to have been included:-

Items inadequately or confusingly explained (please quote page Nos.):-

Items you disagreed with (please quote page Nos.):-

Items you couldn't locate via the Index:-

Any other comments:-

*** Many thanks for your comments above ***