Motion Merging And Other Single Axis Motion Control Challenges

J. Randolph Andrews
Douloi Automation, Inc.
3517 Ryder Street
Santa Clara, CA 95051

Abstract- *Single axis motion is often regarded as the simplest control case. However challenging single axis problems are found in many industrial applications. Challenges include motion merging as well as electronic gearing with boundary cases. Motion merging involves matching speed to a high inertia, free-moving mass which is not being controlled, acquiring control, and realizing a destination position with minimum force disturbances. Electronic gearing appears straight-forward until the boundary cases of velocity and position limits are considered. These example problems become challenging because of interdependence between master and slave motion and transitions during motion. Solutions are presented along with a discussion of controller attributes that simplify implementation.*

I. INTRODUCTION

Although the statement of a motion control requirement may appear straightforward, there are often additional complications brought about by boundary issues and interdependence. Three case studies will be explored to understand what additional complications can occur in the areas of electronic gearing and motion merging and how these issues can be resolved.

II. ELECTRONIC GEARING WITH POSITION LIMIT CASE STUDY

Electronic gearing is generally used to describe a ratioed relationship between two motions. There is conceptually an input to the relationship, labeled a master, and a responsive output motor, labeled the slave. Usually the master is not under the control authority of the motion controller but is an independent motion, such as a "master line shaft" in a large assembly or packaging machine.

The principle equation for electronic gearing appears straightforward:

```
SlavePosition:=MasterPosition*GearRatio
```

This relationship is depicted in Figure 1.

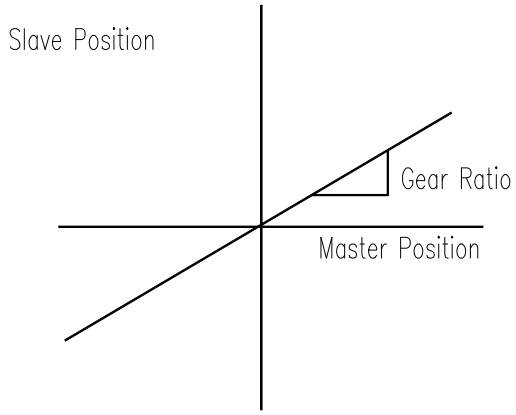

Figure 1. Initial Gearing Relationship

This simple relationship is complicated by practical considerations. The first example is a manual control for a grinding machine table. In this case, the master is manually driven handwheel. The objective is to recreate the familiar crank/motion relationship that is present on a non-automated, manually driven machine tool. The principle equation can be implemented with the following software statements:

```
Procedure EngageHandwheel;
    begin
    while true do
        begin
        Slave.SetCommandedPosition(round(
            Master.EncoderPosition*GR));
        yield;
        end;
    end;
```

What's the first problem with this implementation? The first problem is a discontinuity when this procedure starts. There's no assurance that the initial position of the slave is at the ratioed master position. The first commanded setpoint calculated could be very far from the current slave position. This would cause a large following error and trip the error limit shutdown.

Since we are working with a handwheel, we have freedom to set the position of the handwheel master axis to a convenient location. This would not be an option with the line-shaft scenario. The correction to the first problem is to provide a Slave Offset, or Y intercept as shown in Figure 2.

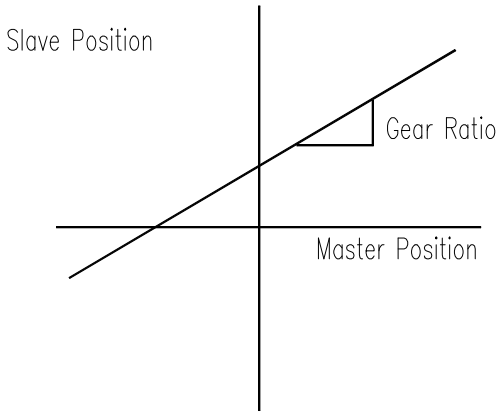

Figure 2. Gearing with Slave Position Offset

This gearing relationship is implemented with the following procedure.

```
Procedure EngageGearing;
    begin
    Offset:=Slave.CommandedPosition;
    MasterAxis.SetEncoderPosition(0);
    while true do
        begin
        Slave.SetCommandedPosition(
            Offset+round(
                Master.EncoderPosition*GR));
        yield;
        end;
    end;
```

What's the next problem with this implementation? If driving a free-spinning motor shaft, this gearing equation works well. But problems occur driving a high inertia load. In this application the grinding wheel table being driven weighs many hundreds of pounds. A small handwheel can be spun, stopped, and started almost instantly with the flick of a finger. The slaved grinding table cannot be because of the inertia. In this case, tracking every slight nuance of the handwheel motion allowed the operator to create high acceleration movement and large forces in the transmission.

One approach to solving this problem is a low pass filter on the handwheel motion. The low pass filter was implemented with the following approach.

```
Procedure EngageGearing;
    begin
    Offset:=Slave.CommandedPosition;
    FilteredPos:=Offset;
    Master.SetEncoderPosition(0);
    while true do
        begin
        ProposedPos:=Offset
            +round(Master.EncoderPosition*GR);
        FilteredPos:=
            (Kf*ProposedPos)+
            ((1-Kf)*FilteredPos);
        Slave.SetCommandedPosition(
            round(FilteredPos));
        yield;
        end;
    end;
```

This routine records the Y intercept as before and proposes a position based on the master position. This position is then filtered by a weighting term Kf which must be between 0 and 1. A value of 1 would eliminate the filter, and a value of 0 would eliminate the incoming information. Practical values for Kf are in the range of 0.1 The slave commanded position is then set to the filtered position.

In this particular application, the range of motion of the grinding machine table is limited. How can position limits be incorporated into the gearing? Several behaviors could be specified. If the handwheel was cranked beyond the table range of motion, the condition could be considered a fault and the motor turned off. Alternatively, the motor position could "saturate" at the limit even though the handwheel traveled further. If this is chosen, what happens when the handwheel reverses? Should the handwheel "unwind" back to the point where table movement stopped, or should the table respond to reversed motion immediately?

In this case, it was chosen to have the slave position saturate and have the handwheel behave like a "slip-clutch". Additional handwheel motion beyond the limit is ignored, and motion resumes immediately on handwheel reversal. This behavior is shown implemented below.

```
Procedure EngageGearing;
    begin
    Offset:=Slave.CommandedPosition;
    FilteredPosition:=Offset;
    Master.SetEncoderPosition(0);
    while true do
        begin
        ProposedPos:=Offset
            +round(GR*Master.EncoderPosition);
        PositivePoint:=Slave.PositiveLimit;
        NegativePoint:=Slave.NegativeLimit;
        if ProposedPos > PositivePoint then
            Master.SetEncoderPosition(
                round((PositivePoint-Offset)/GR))
        else if ProposedPos < NegativePoint then
            Master.SetEncoderPosition(
                round((NegativePoint-Offset)/GR))
        else
            begin
            FilteredPos:=(Kf*ProposedPos)
                +((1-Kf)*FilteredPos);
            Slave.SetCommandedPosition(
                round(FilteredPos));
            end;
        yield;
        end;
    end;
```

Generally masters are independent and slaves are dependent. However, in this case, static position limit information about the slave is used to alter the Master behavior by position limiting the master.

What are the disadvantages of this approach? One disadvantage is that the table comes to an abrupt stop when the slave end of travel is reached. The client judged that since handwheel motion was generally slow and end of travel a relatively rare event, this would not be a problem.

III. ELECTRONIC GEARING WITH VELOCITY LIMIT CASE STUDY

The next case study concerns a remote control camera with panning motion controlled by a servo motor. This application also begins as an apparently simple electronic gearing application, but the specific application requirements direct this behavior in a very different direction than the gearing behavior needed for the grinding table.

A trackball style input was used as a master input with the panning motor acting as the slave. For the same reasons as the grinding application, a low pass filter was found necessary to improve smoothness during operation. Smoothness is particularly important because any jitter is amplified by the radial distance to the camera subject.

After preliminary gearing it was necessary to support changing the gear ration on the fly. A digital input was used to signal that the gear ratio should increase, and another to signal that the ratio should decrease. While moving the trackball back and forth the operator uses buttons tied to these digital inputs to tailor the ratio to the current viewing distance and desired feel.

The apparent solution is to simply alter the gear ratio as a function of the time the two inputs are selected. However changing only the ratio rotates the line in Figure 3 about the Y Intercept and will not work.

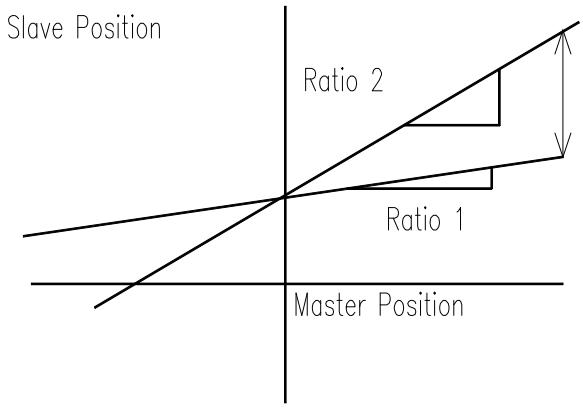

Figure 3. Discontinuity with ratio-only change

If the ratio changes in one sample period by a non-incremental amount, the slave position would jump in a discontinuous manner. In order to achieve position continuity, it is necessary to move the y-intercept. In this case, the master/slave line pivots about the current master/slave position rather than the Y axis. Using this technique, a series of incremental ratio changes would appear as shown in Figure 4.

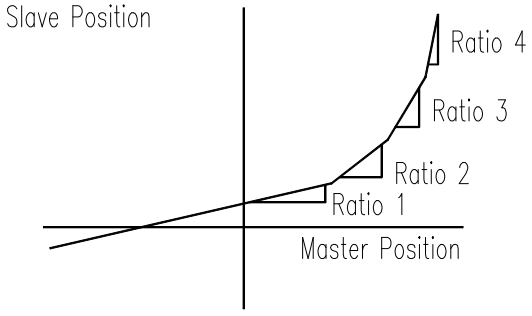

Figure 4. Variable Ratio Approach

An additional requirement of the camera panning application involves managing a velocity limit in the camera pan motor. Because the ratio is adjustable, it's possible to select a high ratio and produce handwheel expressed velocities that can't be obtained by the slave. What's the appropriate behavior if the slave can't keep up? In this case, the choice was made for the slave velocity limit to be converted to a master speed limit through the current gear ratio. During a slew limit situation, the master handwheel position is continually modified to produce the required slippage. This is shown in Figure 5.

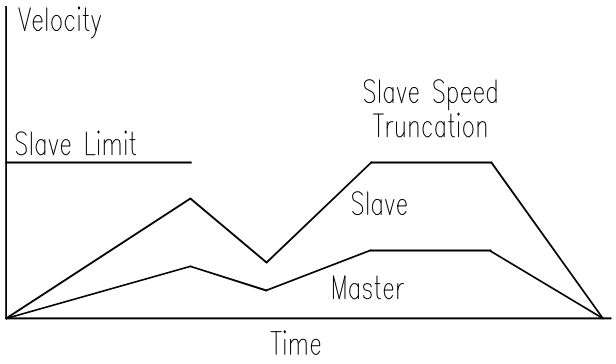

Figure 5. Slave Speed Truncation

The program fragment that implements this slippage is shown below.

```
ActualSlaveDelta:=ProposedSlavePosition.X
    -LastSlavePosition.X;
if abs(ActualSlaveDelta) > MaxSlaveDelta then
    begin
    if ActualSlaveDelta > 0 then
        begin
        Adjustment:=
            (ActualSlaveDelta-MaxDelta)/GR;
        Master.SetActualPosition(
            Master.ActualPosition-Adjustment);
```

The actual slave position change for a specific sample is calculated. If that representation of speed is too large, then a ratioed adjustment is made to the master so that when the master position is used in the next step, it will be within the slave's capabilities.

This is another case where the master is not completely independent but instead is influenced by the static slave attribute of a slew limit.

IV. MOTION MERGING CASE STUDY

The last case study is similar in structure to the rodeo event called calf-roping. The event begins with a calf darting out of a stall. The calf is pursued by a cowboy on his horse. Initially, the calf is in the role of the master, directing the cowboy. The cowboy matches the calf's speed and throws a rope loosely around the calf's neck. After establishing the "linkage" between the cowboy and the calf, the cowboy takes on the role of master in the relationship. He tightens the rope so as to make a low-compliance connection between himself and the calf. The cowboy then dismounts from his horse and decelerates stopping the calf through the linkage. This relationship is depicted in figure 6.

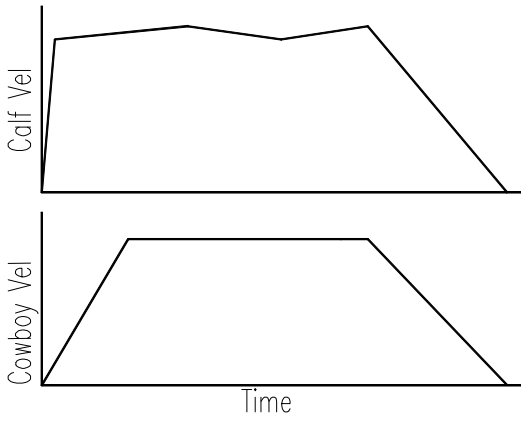

Figure 6. Cowboy and Calf Velocity Profiles

The "calf" velocity is shown in the top curve and the "cowboy" velocity is shown on the bottom curve. The calf takes off at high acceleration. The Cowboy accelerates and matches speed. The cowboy then apprehends the calf and slows down both. Note that the calf might not be holding a steady velocity as the Cowboy tries to match the speed.

The actual piece of equipment being controlled is a forging press. A section of the machine is being back-driven outward by actions of the press. This moving section is like the runaway calf. An attached servo motor plays the role of the cowboy which must apprehend the machine section, slow it down, and stop it. An additional requirement beyond just stopping is stopping at a particular point.

As is characteristic of these case studies, there is an initial approach which falls short and requires some modification to solve practical aspects of the problem. The approach taken was to mimic the role of the cowboy. A position controlled computational, but non-physical, virtual axis chases after the machine section matching the measured speed of the back-driven machine section. When the speeds match, a one-to-one electronic gearing link is established between the virtual axis and the physical servo motor merging the two together to produce a commanded trajectory for the physical servo motor. The motor is then turned on. The virtual axis is then told to stop. This causes the physical servo to also stop through the gearing relationship. Like the rodeo event, there is an initial pursuit, a link established between the master and the slave, and a deceleration of the master causing the slave to stop. This approach was implemented with the following procedure.

```
Procedure EngageOnTheFly;
    begin
    Anchor:=Motor.ActualPosition;
    Delay(100);
    MeasuredSpeed:=
        (Motor.ActualPosition-Anchor)*10;

    Master.Abort;
    Master.SetActualPosition(0);
    Master.SetAccel(10000000);
    Master.SetDecel(Motor..Decel);
    Master.Jog(MeasuredSpeed)
    while Master.ProfilePhase <> 2 do
        yield;
    Offset:=Motor.ActualPosition
        -Master.CommandedPosition;
    Motor.SetMotor(on);
    BeginTask(TaskAddr(Gearing));
    Try
        Master.MoveTo(Destination-Offset)
    Recover
        Begin
        Master.Stop;
        Master.MoveTo(Destination-Offset);
        end;
    GearingEnabled:=false;
    end;
```

The first part of this procedure measures the actual speed of the motor by measuring position change over a 100 millisecond period. The virtual master is then set to 0 position, high acceleration, and a deceleration based on the capabilities of the physical motor. The virtual master then jogs up to the measured speed. The procedure waits for the acceleration phase to complete and continues after the virtual master is at speed. An offset is then calculated as the position difference between the physical motor and the virtual master. This

is quite analogous to the length of the cowboy's rope between himself and the calf. The motor is then turned on and a gearing relationship initiated between the motor and the virtual master. The virtual master is then told to move to the desired destination minus the offset so as to realize a desired physical motor position. It might not be possible for the motor to reach the destination at the current speed and the requested decel if it is too close to the intended destination. In this case the motor stops and reverse to obtain the required position.

This routine references another routine, Procedure Gearing. This routine is shown below:

```
Procedure Gearing;
    Begin
    GearingEnabled:=true;
    while GearingEnabled do
        begin
        Motor.SetCommandedPosition(
            Master.CommandedPosition+Offset);
        yield;
        end;
    end;
```

This gearing routine is like the previous gearing routine with a gear ratio of 1. Although apparently straightforward, initial testing of this approach did not prove satisfactory. At the point of motion merging when the motor was turned on there was a force disturbance. The merge was not smooth. It was noted that the velocity measurement was an average measurement taken over a period of time (100 milliseconds) and that the instantaneous velocity at the merge point was slower than the average because the back-driven, coasting, system was decelerating. The inertia of the machine section made any velocity mismatch a large force disturbance.

It was judged that there needs to be some compliance that could tolerate a slight velocity mismatch. The approach taken was to turn on the motor at the same point as before, but with 0 servo gain. This corresponds to the cowboy lassoing the calf, but with a large rope opening that initially produced no force influence on the calf. After establishing the link and turning the motor on, the gain was then ramped up to a normal position control value over time. This is depicted in Figure 7.

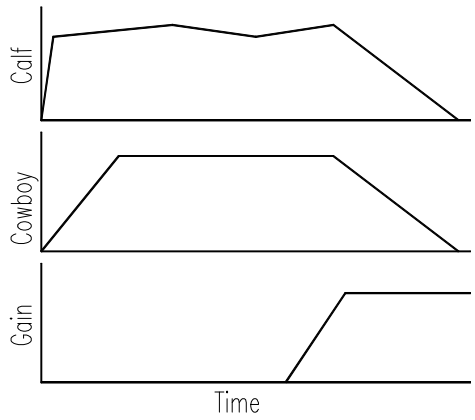

Figure 7. Profiled Gain

This profiling of the gain servo parameter was initiated by the following program lines in the previous procedure, just before turning on the motor.

```
InitialGain:=Motor.Gain;
Motor.SetGain(0);
BeginTask(TaskAddr(CinchGain));
```

Here the gain of the motor is noted, the gain set to 0 and a task started which manages ramping the gain. This ramping of the gain occurs concurrently with the other activities. The ramping is done by the following procedure.

```
Procedure CinchGain;
    Begin
    for scanner:=1 to CinchTime do
        begin
        Motor.SetGain(
            InitialGain*scanner div CinchTime);
        yield;
        end;
    end;
```

This routine smoothly transitions the motor gain from the previously set value of 0 up to the InitialGain setting over a period of time identified as CinchTime. In the actual application, CinchTime is approximately 300 milliseconds as compared to the previous case where the transition was instant. This technique alleviated the force disturbance and is the method of control being used in the forge press today.

This is another case where the master is not completely independent, but is responsive to dynamic information from the slave. The master had to match the speed of the slave prior to engaging the slave. After engagement, the master was able to act in an independent manner.

V. CONTROLLER ATTRIBUTES WHICH SIMPLIFIED IMPLEMENTATION

What are some of the attributes of the control system that allowed these various techniques to be implemented? One helpful attribute is the ability to construct sample rate procedures. In these examples, groups of instructions were performed every controller sample period to achieve custom profiler operations. Controller sample periods are increasing along with improvement in motor performance. Sample rates of 4 kHz are now common. These sample rates only provide 250 microseconds for control law, profiling, and custom sample rate application software to perform.

The control system being considered must be able to insure that all of the required calculations can be performed within the sample period. For example, in the course of adjusting the gear ratio on the fly, it is necessary to change the ratio and as well to change the Y intercept of the gearing in one sample. If only half the calculation is performed, the state of the controller is left incomplete and incorrect for that sample creating a position discontinuity in the slave.

Another helpful controller attribute is access to all the required information. Some control systems provide specific “black-box” built-in behavior. In these examples it was important to have a controller flexible enough to describe a chosen behavior rather than being restricted to a fixed built-in behavior that didn't meet the application need. This flexibility was enabled by having access to all of the required information, not just the ability to call a built-in routine.

VI. CONCLUSIONS

Control problems are seldom as simple as they appear, and even single axis applications can be challenging. Complications can arise because of boundary cases and transitions in control modes while in motion. Controllers which support high speed application programs and which provide access to detailed controller state can be flexible enough to construct tailored, application specific solutions to these challenging problems.

VII. REFERENCES

[1] Instruction Manual for Motion Server & Servo Application Workbench, Douloi Automation, Inc., Santa Clara, CA March 1999

[2] Andrews, J.R. "Motion Server – A Next Generation Motion Controller Architecture", Proceedings of the Twenty Fifth Annual Symposium on Incremental Motion Control Systems and Devices, 1996, pages 1-8

[3] Meyer, Bertrand: *Object-Oriented Software Construction*, Prentice Hall, New York, 1988

About the Author

J. Randolph Andrews received his B.S. M.E. in 1981, B.S. E.E. in 1981, and M.S. M.E. in 1983 from the Massachusetts Institute of Technology. Andrews spent 4 years at Hewlett Packard's corporate research laboratory in the Applied Physics Research Center as well as the Manufacturing Research Center. The following 4 year period was spent with Galil Motion Control. In July 1991 Andrews founded Douloi Automation, Inc. to provide motion control hardware and software solutions for use with Microsoft Windows and Windows NT. Professional interests include motion control, software/electrical/mechanical system design tradeoffs, high abstraction programming, real-time programming, and visual programming techniques and tools.

Paper Presented at the 2000 Incremental Motion Control Systems and Devices Symposium.