

# Controller Requirements for High-Precision Motion Applications

Curtis S. Wilson  
Vice President of Engineering and Research  
Delta Tau Data Systems, Inc., Chatsworth, CA

## Introduction

High-precision motion control applications put demands on a motion controller that many other applications do not. Many of the requirements are subtle, and not immediately obvious from data sheets. The designers of motion controllers must take special care to handle numeric issues, data rates, interpolation strategies, fast servo updates, and many other issues properly if they are to succeed in properly controlling a precision application. This paper surveys the key strategies required for success.

## Commanded Trajectories

Proper strategies for commanded trajectories are vital for the success of high-precision motion applications. This subject is ignored in virtually all academic controls courses, but it is at least as important as the servo algorithms used.

Traditionally, commanded trajectories used a resolution no finer than the feedback resolution itself. In days where every byte of storage and computation was precious, this strategy made sense. It does, however, impose some serious performance limitations. The ideal intermediate points in most trajectories are not in whole numbers of “counts”. The quantization noise introduced by forcing each point to the nearest integer count can be significant.

Even worse are the effects if feedforward is used. The most common feedforward terms, velocity and acceleration feedforward, require digital differentiation of the commanded trajectory – a noise-producing operation. Without substantial fractional resolution, these terms cannot reasonably be used, or used to maximum effect. A lot of the lore about problems with feedforward stems not from the technique itself, but from its use with inadequate resolution.

If positions are generated incrementally – each cycle’s commanded position using the previous cycle’s position as the seed for calculations – substantial fractional resolution is required to minimize numerical integration errors. This is particularly important in multi-axis applications to preserve proper coordination, which often means path accuracy, between axes.

Some of the most careful attention to numerical techniques must be paid in the area of trajectory generation. While many believe that floating-point math is superior to fixed-point math in all aspects, this is not necessarily true. If positions are represented as floating-point numbers, the amount of fractional resolution changes with the position. At a million counts – 100mm from the origin with nanometer resolution – 20 bits of fraction are lost. At a billion counts – 1m from the origin with nanometer resolution – 30 bits of fraction are lost. It is easy to see that standard 32-bit floating-point formats for position representation are utterly inadequate for precision applications. Care even must be taken with 64-bit floating-point representations here.

Furthermore, because floating-point representations are inexact, round-off errors from repeated addition operations can accumulate to become significant, even in applications where high precision is not required, and the floating-point resolution initially appears to be sufficient. If this is a problem in an application, a simple substitution of multiplication for some addition operations can fix the problem.

Virtually all controllers generate their commanded position trajectories using polynomial equations. As controllers have improved, the order of these equations has increased. The earliest digital controllers in the 1970’s used first-order position trajectories, which could generate constant-velocity segments, but not control acceleration and deceleration. These systems required substantial lag in the analog servo drives to create physically realizable trajectories.

The second-order trajectories that became common in the 1980’s permitted basic acceleration control and reasonably smooth cornering ( $C^1$  continuity). This in turn permitted tighter servo control because the need for servo

lag was drastically reduced. Still, these requested step changes in acceleration, and hence in force and motor current, and step changes in curvature, none of which are truly physically realizable.

The third-order trajectories that some controllers introduced in the 1990's permitted jerk control and continuity in path curvature ( $C^2$  continuity). With these, features such as "S-curve" acceleration and cubic splines could be implemented, to powerful effect.

The use of higher-order equations that can generate physically realizable trajectories is particularly important in precision systems that aggressively use feedforward to minimize tracking errors. Systems with substantial tracking errors, or "servo lag", can count on the lag to smooth out sudden command changes, but at the cost of accuracy.

Techniques such as S-curve acceleration are, in effect, low-pass filters on the command trajectory. These serve to prevent the excitation of higher-frequency resonances in the system. A newer technique is to build a band-reject filter into the commanded trajectory for the selective rejection of resonant frequencies, without also rejecting higher frequency content that can provide quick action.

It is still very common to reconstruct contours by commanding points directly on those contours with trajectories (such as "linear interpolation") that inherently fall to the inside of the curve. In doing so, errors are created in the commanded trajectory that dwarf other errors in the system. A modern precision controller must be able to support more accurate contouring trajectories, such as Hermite splines, to minimize these errors.

## **Feedback Issues**

In high-precision applications, the controller must be able to accept feedback signals at data rates (expressed in LSBs per unit time) far exceeding typical motion control applications. The strategies this requires, both in hardware and software, can be different from those for standard control algorithms.

The first and simplest strategy is simply to use fast digital interface circuitry. While a standard quadrature encoder is hard-pressed to put out 500 kHz edge (count) rates, the incremental outputs from precision sensors can often put out 20+ MHz edge rates. Precision controllers must accept these rates.

Many precision sensors, such as interferometers, output a parallel digital data word representing position (or at least distance from a starting point). A precision controller must have the ability to latch a stable data word synchronously with the control-loop execution. If the controller has the ability to perform "software extension" of the position data, then it can interface to only a less significant portion of the parallel word, saving complexity and money.

Increasingly, controllers are asked to accept directly the "sine-cosine" analog signals from interferometers, linear scales, and rotary encoders, and resolve them with a high degree of interpolation. This interface typically requires a hybrid analog/digital interface, with a digital cycle counter and a pair of analog-to-digital converters combined with an arctangent algorithm (in hardware or software) to determine where within the cycle the system is. Good algorithms to combine the integer and fractional components of the cycle count without any transition glitches are difficult, but essential.

## **Servo Algorithms**

Perhaps surprisingly, most high-precision motion applications do not require complex servo algorithms. Most of these systems engineer their mechanical systems so well that they do appear to the controller as simple rigid bodies; if hydrodynamic bearings are used, the problem of non-linear mechanical friction disappears as well. Standard-format PID controllers do very well on this type of system.

What the servo algorithms for these systems do require is a lot of speed and a lot of resolution. In the vast majority of industrial digital control systems, what limits the gains of the servo loop is not the classic stability criteria, but delays and quantization noise due to limited resolution. Delays, which add destabilizing phase lag to the loop, come first from the digital sample rate, which automatically introduces a half-cycle delay, and if velocity information is derived from position feedback, from the digital differentiation for the feedback, which introduces another half-cycle delay.

Further delays come from the finite calculation time between when the feedback is read and the command effort is computed, and from the “transport delays” required to get the computed value(s) converted to actual physical signals to drive the actuator. Many modern “smart” drives – digital drives with analog inputs – have exacerbated this problem by compounding these delays. A digital controller samples the position periodically, closes the position and possibly velocity loop, clocks the data to a digital-to-analog converter so the command is sent to the drive in analog form. The analog-to-digital converter in the drive periodically, but asynchronously to the controller, samples this signal, closes the current loop and possibly the velocity loop, then converts the voltage command effort values to physical signals. Multiple sample cycles of delay build up through this process, drastically reducing the gain, and hence the performance, of the system.

It is vastly superior from this point of view to close all of the loops in a single processor and quickly convert the command output values to physical signals (usually PWM on-off signals to the power transistors). This fully digital control scheme – including digital current-loop closure – can provide greatly superior gains, resulting in better stiffness, disturbance rejection, bandwidth, accelerations, settling times, and other attributes.

The finite resolution of digitized feedback also limits gains. Providing higher feedback resolution to the servo loop, particularly if the numeric values are differentiated (as in a velocity loop from position feedback), can substantially improve system performance, even if there are not corresponding increases in repeatability and accuracy. In one test, a common physical system consisting of a 1 kW brushless motor with a 1000-line sinusoidal encoder directly driving a ballscrew with a 10 kg load was tested with various degrees of interpolation on the encoder. The test move was a 25 mm move, settling to 1 micron. The results were:

- “x4” decode (4000 states per rev): 100 msec
- “x256” interpolation (256,000 states per rev): 75 msec
- “x4096” interpolation (4,096,000 states per rev): 60 msec

Elimination of mechanical friction through tools such as air bearings can be very important in precision applications to avoid the issues of “stick-slip” oscillations, but by eliminating the damping effect provided by the friction (even if it is non-linear), stability issues are made worse for the controller. A good rule of thumb is that the servo loop must be closed at twice the frequency of a comparable system with a little mechanical friction. As a practical matter, the servo update rates for these systems are a minimum of 4 kHz.

Feedforward terms as part of the servo algorithm can be critical in a precision system to keep tracking errors to a minimum, not waiting for errors to build up for the feedback algorithms to react. Luckily, high-precision mechanics are particularly amenable to the aggressive use of feedforward, with their dynamics typically being quite simple and readily modeled. In such a system, over 99% of the servo command can come from the feedforward terms, with the feedback algorithms just providing the occasional correction for disturbances and modeling errors.

## **Commutation Algorithms**

In many applications that require limited precision, a simple switching commutation strategy for brushless motors, commonly known as “six-step” commutation, can be employed. In high-precision applications, more sophisticated commutation algorithms, commonly known as “sinusoidal” commutation, almost invariably must be used.

Even with “trapezoidally wound” brushless motors, six-step commutation provides significant torque/force ripple over the span of a single step, and the sudden switching of current in phases at step boundaries produces a noticeable torque/force perturbation. This can result in visible marks in the surface finish of a cut part, for example.

Because the torque produced by a commutation algorithm is proportional to the cosine of the measurement error in the rotor’s magnetic field angle, small measurement errors, such as those due to resolution limitations, have little effect on the results of the algorithm. Therefore, it is not necessary to have as much resolution for the commutation algorithm as for the servo algorithm. Typically, however, the commutation algorithm uses the same sensor as the servo algorithm. All of this resolution may not be used; it is common to use the closest point in a sinusoidal look-up table of more limited resolution.

The strategy for closing the current loop is inextricably tied up with the commutation algorithm. Microprocessors, particularly DSPs, have recently gotten fast enough to perform current-loop closure digitally. This permits several

important improvements. First, analog transmission of commands, which are noise-sensitive, can be eliminated. Second, it is possible to do all of the feedback calculations, for position loop, velocity loop, commutation and current loop, in a single processor, minimizing transport and calculation delays, which, as explained above, leads directly to higher possible gains and performance.

### **Compensation Algorithms**

No physical system is perfect, and it is often more cost-effective to measure these physical errors and compensate for them algorithmically than to fix them physically. A good precision controller needs a variety of techniques to compensate for errors. These usually include position (“leadscrew”) compensation tables with independent corrections in the positive and negative directions, two-dimensional and sometimes even three-dimensional compensation tables, and squareness correction algorithms.

At the highest levels of precision comes the realization that even the best systems that are supposedly Cartesian are never truly Cartesian, especially as temperatures vary. Advanced users are now experimenting with the use of robotic-style kinematic algorithms to handle these slight deviations from the ideal Cartesian systems.

### **Synchronization Issues**

In these applications, the motion itself is really only a means to an end, with the motion required to create a certain spatial relationship for an operation. In many applications it is desired to perform an operation during rapid motion with as tight a relation as possible to the physical position. Two strategies that can assist in this are “position-capture” and “position-compare” circuits.

Traditional controllers have performed these functions in software, limiting the accuracy of these functions due to the latency of the software loop. Around 1990, controllers started to appear with dedicated digital hardware for the capture and compare functions, eliminating software delays. The transition of a digital input signal (encoder index, probe switch, proximity sensor) immediately “captures” the present encoder counter value into a dedicated register, independent of where it is in the servo cycle. The software can then process this information later without its delays hurting the accuracy of the process.

The position-compare process is logically the inverse of the capture. The processor pre-loads a hardware register with a value equal to the encoder count at which an output signal is desired. Without further software intervention, the digital hardware “continually” compares this value to that of the encoder counter. When the two values are the same, a digital output value is immediately triggered.

These digital-hardware techniques revolutionized the synchronicity of motion to digital inputs and outputs. However, as demands for higher resolution have taken the processing of feedback past simple digital quadrature into more complex hybrid analog-digital processing, the capabilities of the digital capture and compare techniques have not kept pace with the increased resolution.

A new technique rectifies this imbalance. The digital hardware associated with the encoder counter uses dedicated timer circuits to compute fractional count values at the same rate at which the counter can increment. With updated fractional count information now always available, the capture and compare circuits can be extended to incorporate the fractional information as well. When used with interpolation circuitry for sinusoidal encoders, as long as the zero-crossings of the sine and cosine waves are used internally to feed a counter associated with the timers and capture/compare circuitry, the resolution of these circuits can “catch up” to that of the interpolator.

(Note that earlier timer-based strategies for sub-count interpolation only calculated the fractional count information once per servo cycle – inadequate for immediate capture and compare functions.)

Because the zero-crossings of sinusoidal encoders are their most “dependable” feature, this purely digital timer-based circuitry is not only simpler and less expensive than analog-based capture/compare circuits, it is at least as accurate as well.