DELTA TAU
Data Systems, Inc.

Control Requirements for High-Precision, High-Speed Machining

Curtis S. Wilson
Delta Tau Data Systems, Inc.
Northridge, California

INTRODUCTION

It was not too long ago that machine tool builders had to choose between constructing a machine capable of great precision or one capable of great speeds. Increasingly machine tool users are demanding both capabilities in the same machine, forcing builders to push the envelope on machine capabilities. These new types of machines put increasing demands on the controller. This paper explains some of these demands in the areas of feedback data rates, motor phase commutation, servo algorithms, and trajectory generation.

FEEDBACK DATA RATES

Early in the analysis for a high-precision, high-speed machine, the designer will come across the problem of feedback data rate. Simply put, the required feedback data rate is the product of the sensor resolution and the maximum speed. In whatever form the data is presented, the controller must be able to accept data at this rate. For example, on a machine with 0.1 μm (4μin) resolution and a top speed of 6 meters per min (0.1 mps; or 240 ipm, 4 ips), the controller must be able to accept 1 million counts per second.

Pulse Counting

If the data is presented in the purely incremental form of a pulse train, usually in quadrature format, then for the controller the maximum rate is simply a question of counter speed. In these applications there must be a hardware counter, and the question is fundamentally dependent on the maximum clock frequency at which the counter can be run. A digital counter cannot accept more than one count per clock cycle. If a second edge comes in during the same clock cycle, it is lost to the counter, and permanently.

So at the very minimum, our example system that must accept 1 million counts per second must have a 1 MHz clock for the counter. But pulse trains are never perfectly even, so it is best to degrade the maximum count rate at least 20% from the clock frequency, even for good encoders. With this, our required clock frequency is at least 1.25 MHz. Furthermore, if the controller uses one of the increasingly popular digital delay filters that filter out noise spikes up to one clock cycle wide, the clock frequency should be doubled (in our example, to 2.5 MHz); with a filter that blocks noise spikes up to two clock cycles wide, the clock frequency should be tripled.

Parallel Word Read

When the data is presented to the controller in parallel form, as from an absolute encoder, or from many laser interferometers, the key controller limitation is a bit different. If the parallel word covers the entire range of the axis, there is no controller limitation on data rate; the axis can be at one end one instant, and at the other end the next. But providing this many data lines to the

controller is usually far too expensive; if our machine had a 1 meter travel, 24 bits would be required.

Most machines with parallel data feedback provide only a limited number of lines back to the controller. When the data on these lines rolls over, it is the responsibility of the controller to extend the position in software. In order to do this properly, it must sample the data more than twice per cycle; that is, the controller cannot permit half a cycle of the input data to pass between samples. Readers familiar with digital sampling theory will recognize this as an example of the Nyquist Sampling Theorem, although perhaps in an unfamiliar context.

In our system with 0.1 μm resolution, if 12 bits were presented in parallel to the controller, and the controller sampled the data at a 1 kHz rate, the maximum speed properly measured by the controller would be about 2000 LSBs per sample, or 200 μm/msec, or 200 mm/sec.

In controllers with hardware counters, the counter appears to the controller as parallel data, and it is rare the counter covers the entire range of the system. Therefore these systems also have this Nyquist limitation. However, because most of these counters have 16 or 24 bits, it is rare that this speed limit is lower than the counter clock rate limitation or the sensor frequency limitation. With a 16-bit counter and a 1 kHz software sample, our system would have a 3200 mm/sec speed limitation here. In older systems with 8-bit counters, this could be a real limitation; in our system, this would yield a top speed of about 12 mm/sec.

Electronic Sensor Interpolation

Increasingly, manufacturers of incremental sensors are turning to electronic interpolation techniques to increase the resolution of their sensors past the traditional 4 counts per sensor "line" (wavelength in an interferometer). Electronic interpolation is proving more cost effective than increasing the physical resolution of the sensor. These techniques force the question of how to present this high-resolution data to the controller.

Linear scale manufacturers have typically produced a higher frequency quadrature waveform in order to be able to interface with existing controllers. However, the higher frequency can often outrun the decode and counter circuitry in the controller. In many cases, movement can outrun the interpolator's ability to output pulses, causing a saturation "burst mode", until the interpolator's tracking loop catches up with the real movement. If this occurs, the controller is not getting accurate position information during the burst mode.

Interferometer manufacturers generally opt for parallel-word output format, because the required quadrature frequency would be too high in most applications. Typically 32 bits of position information are provided, although a controller that can handle rollover and software position extension does not require the use of this many bits. Still, the parallel data format can be cumbersome and expensive for interfacing.

A relatively new interface format for both encoders (scales) and interferometers combines a digital quadrature signal at or near the fundamental frequency of the sensor, combined with a short parallel word containing the fractional position information. This hybrid format offers nice tradeoffs between maximum speed and numbers of lines, keeping the quadrature frequency within the range of most counters, and keeping the number of data lines reasonable.

MOTOR PHASE COMMUTATION

High-speed, high-precision applications can put demands on the commutation subsystem that do not exist in many other applications. The brush motors that are used in many applications are often not acceptable here. The stick/slip friction between brush and commutator bar can make it very difficult to hold tight position tolerances. The nature of brush commutation makes it difficult, if not impossible, to correct for the torque ripple introduced. Also, the limitations in power that can be passed through the commutator put additional limits on the motor torque/speed curve. Finally, brush motor armature windings are on the rotor, where their heat cannot be well isolated from the parts of the machine that must be kept thermally stable for maximum accuracy.

For all of these reasons, electronically commutated brushless motors are now very popular in these applications. But the nature of the brushless motor and the method of commutation are still important.

"Six-Step" Commutation

The early "trapezoidally wound" brushless motors and the matching "six-step" commutation algorithms working from hall-effect sensors, while suitable in many applications, fall short in many high precision applications. The algorithms introduce significant torque ripple as a function of motor angle, even statically. Furthermore, because they switch current among the phases of the motor rather than apportioning it, the switching transients introduce potentially noticeable disturbances. Sometimes the six switching points of the motor can be seen as imperfections in the surface finish of the cut part!

Sinusoidal Commutation

Increasingly, "sinusoidally wound" brushless motors with sinusoidal commutation algorithms are being employed. In the ideal case of a motor with truly sinusoidal phase torque/back-EMF curves and no cogging torque, the result is zero torque ripple. Take the example of a two-phase motor (the principle extends to higher numbers of phases) with torque-angle functions:

$$T_a = K_T I_a \sin\theta$$
$$T_b = K_T I_b \cos\theta$$

If we provide current to these phases according to the functions:

$$I_a = K_I \sin\theta$$
$$I_b = K_I \cos\theta$$

then we get a total torque from the motor:

$$T = T_a + T_b = K_T K_I (\sin^2\theta + \cos^2\theta) = K_T K_I$$

which shows no variation as a function of angle. Of course, no real motor attains this ideal, but good design can bring the torque ripple down to 1 or 2%.

What of the imperfect motor? Does the ripple introduced by imperfections in the design simply have to be accepted? Of course not! If the nature of the imperfections is known, compensation is possible, and perhaps even straightforward in a sophisticated digital controller. "Sinusoidal" commutation is virtually always implemented using a lookup table for reasons of computational speed, and there is no inherent reason why the table cannot be changed to a shape other than a sinusoid.

Resolution Requirements

Obviously, sinusoidal commutation requires more resolution in the commutation sensor than does six-step commutation. Fortunately, the servo loop's position sensor in a precision system has more than enough resolution for the commutation algorithm. This sensor can then be used for both algorithms.

How much resolution does the commutation algorithm need in a precision system? Basically, enough resolution to make the algorithm's contribution to torque ripple negligible compared to other factors. This requires less resolution than many people suppose. Take a system that uses a 256-point lookup table, for a spacing of 1.4^{o}. Intermediate positions are rounded to the nearest point in the table. The reduction in torque introduced in this rounding is proportional to (1-cos ε), where ε is the difference between the actual position and the position used in the algorithm. The worst roundoff in this system is 0.7^{o}, resulting in a torque reduction of 0.0001, or 0.01%. This is far smaller than the amount introduced by any real motor itself.

Phase referencing

Synchronous motors such as permanent-magnet brushless motors require position information that is absolute at least over one commutation cycle. Many people take this to mean that an absolute sensor is required, but this is not true. An incremental sensor can be used provided that a phasing search move can be done (and completed reliably) every time the machine is powered up. This can save a significant amount of money, particularly on motors, for which absolute sensors are exorbitantly expensive. This type of phasing search usually requires that commutation be performed in the controller, which has sufficient intelligence to perform the search.

SERVO ALGORITHMS

The servo loop algorithms are a critical part of any high-precision system. There are many issues involved in designing or selecting an algorithm, but these issues can generally be divided into three categories: speed, resolution, and structure.

Update Rates

Virtually all controllers today use at least partially digital control algorithms, and "speed" of an algorithm in a digital controller refers to the sample rate or update rate. In general, the faster the update, the better. At higher update rates, disturbances can be noticed sooner, and compensated for sooner. Also, loop gain can be increased more without instability; a higher command can be made for a given level of error without worry of overshoot if it can be turned off more quickly.

High update rates are particularly important on "zero-friction" systems. Mechanical friction causes a host of problems in a high-precision machine, but getting rid of it with fluid or magnetic bearings actually introduces other headaches. The main problem is the loss of the stabilizing effect of the damping that mechanical friction provides, albeit in non-linear fashion. This makes it easier to create oscillation about the commanded point from repeated overcorrection at a given update rate. Loop gain must be kept low to prevent this cycling, which limits stiffness and bandwidth. This leads to a requirement for very high update rates on zero-friction systems.

Position Sensor Resolution

The position sensor's resolution can limit the servo update rate if it is being used to derive velocity information for servo stability, as in a lead filter or PD algorithm. Fundamentally, the velocity estimation includes (even if it is not limited to) subtracting consecutive position readings, a digital differentiation operation that magnifies quantization errors. The higher the update rate, the worse the quantization error problem. Getting damping for the purposes of achieving stability involves multiplying this velocity estimate into the command value, including the quantization errors.

At high update rates, it can be impossible to achieve the necessary damping for stability without getting unacceptable quantization noise from the digital differentiation. Thus, limited sensor resolution can limit the servo update rate, which in turns limits the stiffness of the system.

As the trend toward more fully digital control systems proceeds, this limitation has spawned a lot of work into very high-resolution sensors, to the point where the resolution is sometimes much higher than the *accuracy* needs of the machine require. This ultra-high resolution has in turn pushed the feedback data rate limits of control systems, as described above.

Feedback Algorithm Structure

Perhaps surprisingly, the most advanced control algorithms coming out of university research are not that important in today's high-precision machines. This is essentially because such great care is taken in the mechanical design to get high stiffness and eliminate non-linearites such as friction and motor torque ripple that the machine dynamics are mathematically very simple. Many of the most accurate machines in the world use a simple PID filter, and the tuning of the PID is often not a major factor in the development of these systems.

In the early days of CNC controllers, a simple proportional position-loop gain feeding an analog velocity-mode amplifier was all that the digital controller could master. The stiffness of the system was obtained through high velocity-loop gain. Typically integral gain, or a lag network, in the analog velocity loop was used heavily to obtain this stiffness while working against heavy cutting loads. This can work well when velocities and accelerations stay low, but the lag introduced by integration reacts badly when trying to track significant accelerations well.

When higher accelerations are desired, loop stiffness cannot be achieved through analog velocity loop integration; the digital position loop must provide it. For the reasons shown above, this can require high position sensor resolution, high update rates, and high command output resolution in the controller, all of which have become more common in recent years.

Increasingly, as the digital techniques for velocity estimation improve along with sensor resolution, and as analog velocity integration becomes less important, fewer builders are willing to put up with the cost and inconvenience of analog tachometer-based velocity loops. Instead they use digital velocity loops in the controller, whether simply a derivative gain, or more complex velocity estimators.

These velocity estimators can use supplemental hardware, such as timers measuring the time between encoder transitions -- the velocity is estimated as being inversely proportional to the time, giving these estimators their common name of "1/T". Other estimators use several recent position measurements, output command values, and knowledge of the dynamics of the system to compute a velocity value. The need for knowledge of the system requires a "tuning" of the estimator for a particular machine.

Feedforward Algorithms

Feedforward techniques are becoming increasing popular in machining applications. The idea of feedforward is to anticipate the command needs of the system for the programmed trajectory and inject these values straight into the controllers command output without waiting for errors to build up and the feedback algorithm to create the command. The feedforward estimate will not be perfect, of course, but the feedback algorithm is still there to "mop up" for any errors in the estimate, and to counteract any disturbances.

Analytically speaking, the feedforward algorithm should have a transfer function that is the inverse of the plant's transfer function. In the ideal case, this creates a net unity transfer function between the commanded trajectory and the actual trajectory, which means the actual trajectory follows the commanded exactly.

A well engineered physical plant will usually appear to the controller like a rigid body inertial load with some damping, either from physical sources or from the velocity loop closure. In a traditional machine tool, the controller did not try to compensate for the lags introduced by either the inertia or the damping. The effect of the tachometer based damping was bigger, leading to an error proportional to velocity typically expressed in "inches-per-minute per mil" or "meters-per-minute per millimeter". By the simple expedient of adding directly into the controller's command output an amount proportional to the desired velocity, this error can largely be eliminated. This technique is known as "velocity feedforward".

If the errors of such a system with velocity feedforward are examined, a large component of the error will be proportional to the acceleration. This is due to the fact that inertia resists acceleration. This error can largely be compensated by adding into the command an amount proportional to the desired acceleration, a technique known as "acceleration feedforward".

Many more sophisticated feedforward techniques are possible, but velocity and acceleration feedforward are sufficient in a surprisingly large number of cases. In machining applications, these trajectory-based feedforward techniques are useful primarily when the cutting loads are light, and do not dominate the required forces. This tends to be the case in the new class of high-precision, high-speed machines, at least on the finishing cuts. "Disturbance feedforward" techniques can also be used to cancel out major load disturbances, such as high cutting forces.

TRAJECTORY GENERATION

In these types of applications, it is vital that the commanded paths that we request be both accurate and physically realizable given the dynamics of the system. This is especially true when you cannot count on servo lags to smooth over impossible features in your trajectories. It does no good to generate the world's most accurate commanded trajectory if the machine cannot follow it.

Almost any trajectory generation algorithm will have two parts: trajectory planning, which occurs once per block, and trajectory update, or interpolation, which occurs once per servo update. Sometimes for internal computational reasons there are separate coarse interpolation and fine interpolation routines; the coarse interpolation is executed at a rate between the block rate and the servo update rate.

The purpose of the trajectory planning algorithm is to take the move description from the part program and system constants -- positions, velocities, accelerations, and interpolation mode -- and generate the equations of motion for each axis, usually as a set of polynomial coefficients for the position-vs.-time equation for each axis.

Position Update

The update, or interpolation, algorithms then solve these equations every servo cycle to create the series of commanded positions that are inputs to the servo algorithms each cycle. The order of these equations that are generated and solved is an important consideration in the design or selection of a machine.

First-order position equations are capable of generating constant velocity trajectories, which can produce straight-line paths in a cartesian system. These have been used to create the chordal approximations of contours that are well known. However, these types of systems require either an exact stop at all corners, or considerable lag in the servo to provide proper rounding of the corners.

With second-order position equations, it is possible to control acceleration as well as velocity; each segment has constant acceleration, if not a constant velocity. Because the direction of motion in a cartesian system is directly related to the ratios of the velocities of the axes -- if the ratios are constant, the path is straight -- acceleration control provides the capability to generate smoothly curving paths. This acceleration control is often used to provide smooth blending between straight-line segments, which has the effect of rounding to the inside of the programmed corner. Sometimes the blending becomes the entire move, which creates completely curved paths.

Third-order position equations have recently become common on machine controllers. The extra order permits the control of "jerk" (time derivative of acceleration). Because acceleration requires proportionate motor current, which cannot change instantaneously due to inductance, controlling the rate of change of acceleration important in generating precisely realizable trajectories.

Trajectory Planning

There are many ways the trajectory planner can use the update algorithms. Traditional acceleration control causes rounding of the path to the inside of the programmed points if it is

desired to pass these on the fly. This rounding can be a major source of error in many systems; the closer together these points are, the smaller the rounding error. This has led many systems to use ridiculously high block rates to keep this error within acceptable bounds.

However, this rounding error is very predictable, and a technique that is often better is to pre-compensate for the expected errors. A simple pre-correction that eliminates the overwhelming component of rounding error in a 2nd order blending profile is to adjust the commanded point coordinates P_n for each axis to the outside according to the formula

$$P'_n = (-P_{n-1} + 10P_n + P_{n+1}) / 8$$

A similar correction for 3rd-order splines (B-splines) uses the formula

$$P'_n = (-P_{n-1} + 8P_n + P_{n+1}) / 6$$

The trajectory errors with and without pre-correction are summarized in the following table:

Method	Via Point Error	Error Variation	10^o Segment Error
1st Order (Chordal)	0	$V^2 T_a^2 / 8 R$	$3.8 \times 10^{-3} R$
Corrected 1st Order	$-V^2 T_a^2 / 16 R$	$V^2 T_a^2 / 8 R$	$1.9 \times 10^{-3} R$
2nd Order	$V^2 T_a^2 / 8 R$	$V^4 T_a^4 / 144 R^3$	$3.8 \times 10^{-3} R$
Corrected 2nd Order	$V^4 T_a^4 / 64 R^3$	$V^4 T_a^4 / 144 R^3$	$1.4 \times 10^{-5} R$
3rd Order	$V^2 T_a^2 / 6 R$	$V^4 T_a^4 / 384 R^3$	$5.1 \times 10^{-3} R$
Corrected 3rd Order	$V^4 T_a^4 / 36 R^3$	$V^4 T_a^4 / 384 R^3$	$2.6 \times 10^{-5} R$
Position-Velocity	0	$V^4 T_a^4 / 384 R^3$	$2.4 \times 10^{-6} R$

When executing trajectories at high speeds with a high block rate, it is not possible to come to a stop in the distance between adjacent points. This means that the traditional technique of simply looking for the next point and deciding whether to blend to that point or decelerate to a stop at the latest point will not work. The decision may have to be made dozens of blocks ahead.

The lookahead algorithms that perform this task are becoming an increasingly important feature in modern controllers. Sophisticated algorithms can do the lookahead while executing the program, staying far enough ahead to ensure a controlled stop. Many of these algorithms also calculate centripetal acceleration based on instantaneous radius, and control the velocities to keep the net vector acceleration (tangential and centripetal) within system limits.

CONCLUSIONS

A successful high-precision high-speed machine requires the proper integration of mechanics, electronics, sensors, actuators, and controls. It is important to assess the required control features early on in the design of the machine, and create the proper level and match for servo algorithms, trajectory generation, motor commutation, and sensor resolution. All these factors are interlinked, and limitations in any one area can lead to failure in the machine design.