Eliminating Non-Repeatable Errors

Repeatable error is easy to accommodate and correct.

Repeatable error is easy to accommodate and correct. But speed changes and servo-system bandwidth issues sometimes cause non-repeatable error in automated systems.

Fortunately, some hardware with firmware command sets can make all errors predictable, and therefore manageable, even when line speed varies. It's particularly useful in large roll web processing, because it makes for good plastic, paper, and other sheet product even from the start of a roll (when the motor is moving slowly because of the large circumference of the roll) as well as near the core (when the motor may not be able to keep line speed very high due to the small circumference) — for almost no scrap and much better repeatability.

Q&A

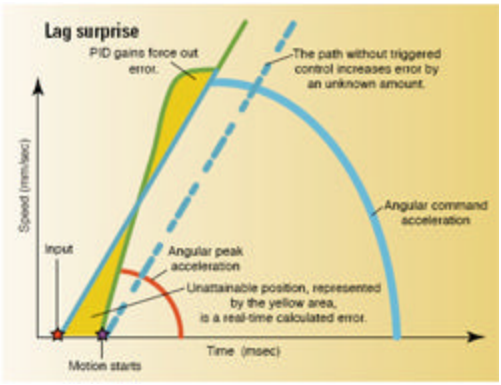
What usually causes non-repeatable error?

A change in process speed. When commanded to move, all servo systems initially do nothing. This is the servo lag present in all systems. Then the system starts to move.

Say a line of paper is going by at 3,000 ft/sec and at every 200 ft, we see a marker. Well, with a servo lag of 500 µsec we can expect 1.5 ft of error. Now 1.5 ft is a large error, but because it is a known, repeatable error, we can anticipate it, and live quite happily making perfectly good product. But what happens if line speed drops by half? Speed is then 1,500 ft/sec, but because lag time for the servo system is fixed at 500 µsec, expected error is now 0.75 ft or 9 in. The good news is that error is smaller than when the system was going faster; the bad news is that error is different by 9 full inches. The inability to compensate for this different, speed-induced error change exposes limitations caused by servo system bandwidth.

How do triggered moves help?

By being fast. A high-speed hardware register records the input in a sub-microsecond timeframe, and then another register records when movement actually begins. The difference between the two is the time period that defines the period where the system should have been moving. Speed over a period of time equals position; system



Triggered gear move command sets allow an axis to accelerate to a master encoder in a fixed length of product, despite encoder speed variances. Triggered cam commands make cams start on a trigger and accelerate to geared line speed — despite varying line speed — to accelerate the cam to line speed over a fixed, known length.

firmware quickly calculates error or unattainable position loss by looking at the commanded acceleration. Time ? t between input and actual motion and the commanded acceleration rate define the unattainable position error caused by lag. This position error is fed the high side of the comparator circuit, and then — after comparison to feedback — is sent as raw error to the position loop. The position loop gains aggressively eliminate this (and all other errors) from the system.

Now, this seems fairly straightforward, but only firmware hardware can make these calculations fast enough. Solving this math and then making an input to servo command over software takes far too much time. Why? Total acceleration time is typically less than half a second. So, hardware registers must be sub-microsecond, and response sub-millisecond, to make calculations based on hard numbers, and give repeatable results. Otherwise, the system would just take the dotted blue path without registering any of the non-repeatable error.

What specific commands measure and solve this error?
Some controllers have triggered gear, cam, and input move command sets implemented with the necessary hardware standard on board. Triggered gear commands allow an axis to accelerate to a master encoder in a fixed length of product — despite encoder speed variances. A triggered cam command ensures that cams start on a trigger and accelerate to geared line speed with a planned distance in product, despite the cam's acceleration up to line speed.

By Steve Reese

Reprinted from *Motion System Design* magazine, January 2008.

www.parkermotion.com