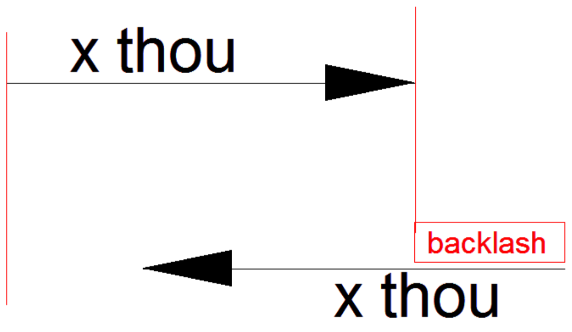
Adaptive Backlash Reduction, Version 1.0

By R. G. Sparber

The idea presented here has ***not*** been brought to a successful completion. My goal is to offer what I have done in the hope that others will make suggestions that will break the "log jam".

The Problem

Especially in hobby grade Computer Numerical Control, backlash is one of the biggest headaches. This is because most of these systems run "open loop". This means there is no way of sensing actual position and correcting errors in movement. The system thinks it has moved the specified distance but some of this movement was used to traverse the gap between moving parts: backlash.



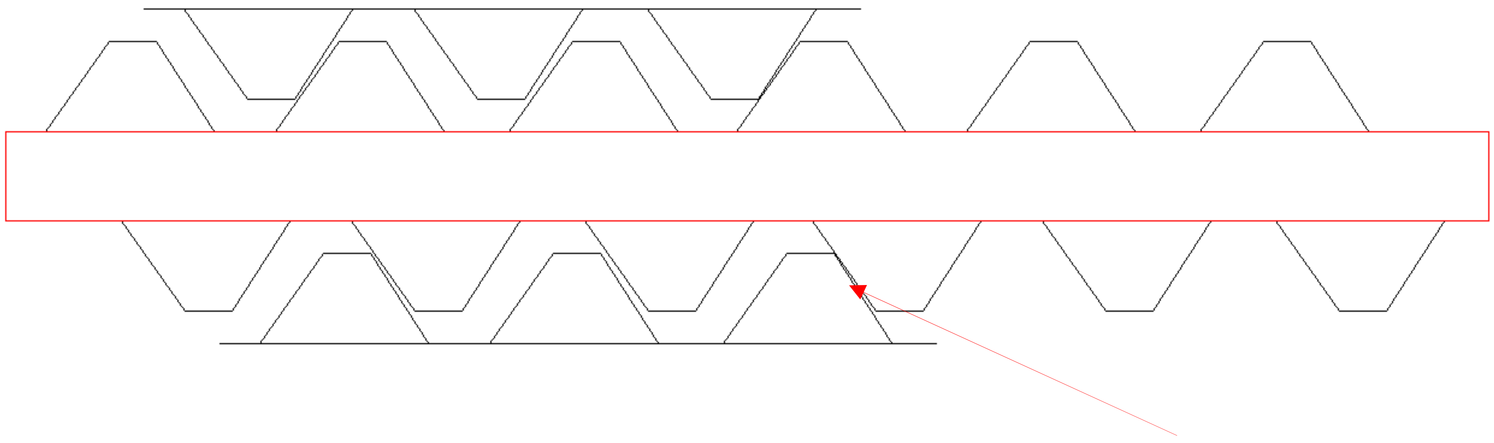
Backlash rears its ugly head when the movement requires a reversal in direction. Ideally, we would want to tighten up the mechanism so backlash goes to zero. But doing so causes binding because of variations in fit. This is especially true between a lead screw and its take up nut[2].

Rather than tightening the take up nut until it almost binds, why not sense the need to quickly traverse the backlash zone? This backlash zone is called the "dead band".

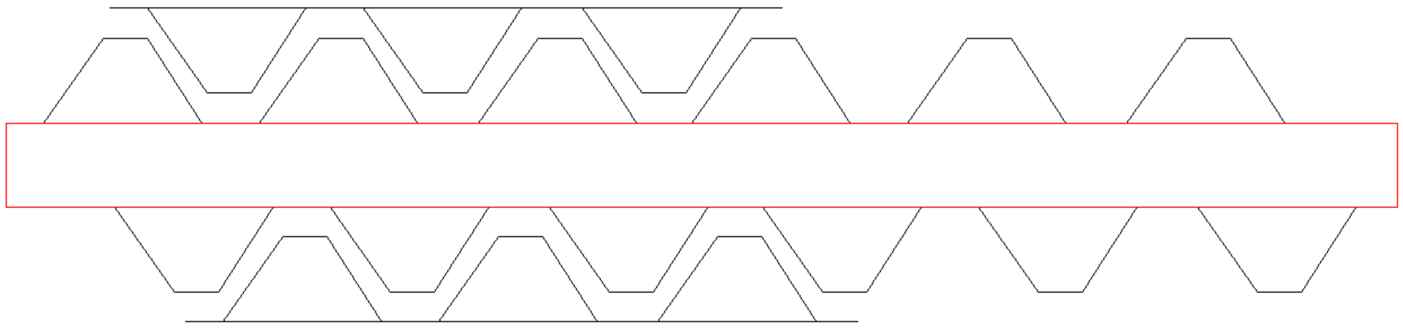
The essence of Adaptive Backlash Reduction is the ability to sense when we are in the dead band and turn the leadscrew until we are back to full contact between leadscrew and nut.

[1] You are free to distribute this article but not to change it.
[2] The problem is much less with reasonable quality ball screws.

Consider the ideal case of a leadscrew and nut where there is perfect alignment but also backlash.



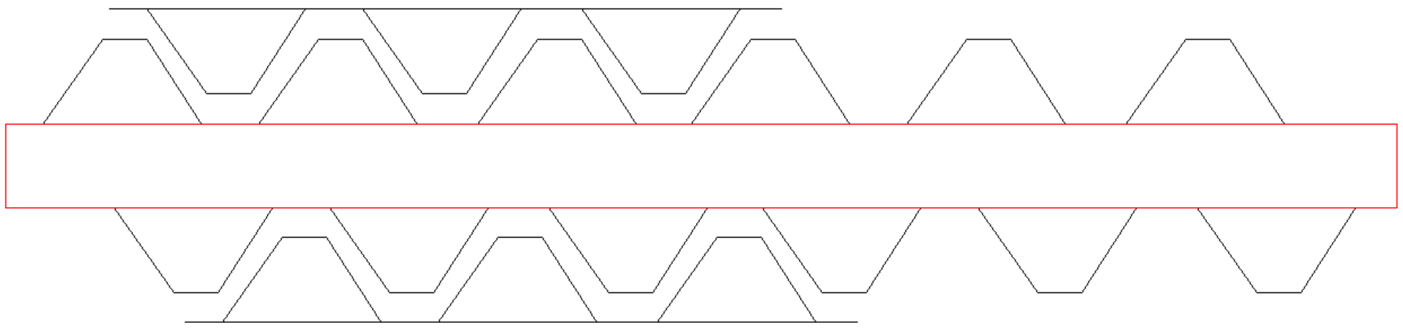
When the leadscrew is engaged with the nut, contact is made on the face of the thread.



When the leadscrew is in the dead band, the leadscrew is not touching the nut.

I am using CNC software that can do backlash compensation. On Mach3, this consists of driving the leadscrew a prescribed distance each time the direction of motion reverses. I just specify the backlash plus the speed of traverse. This speed is a percentage of maximum speed.

If backlash is constant, the software solution works very well. But if backlash changes with distance along the leadscrew, the change in backlash cannot be addressed. Backlash can be a function of the position of the nut along the leadscrew. But it can also be a function of the rotation of the leadscrew. This rotational error is cyclic and can be larger than the positional error.



Now, say the leadscrew was electrically isolated from the nut. I can then detect when I am in the dead band:

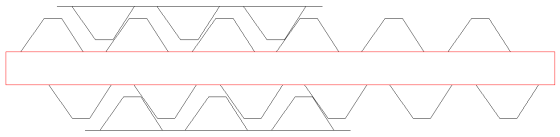
- open circuit means I'm in the dead band
- closed circuit means I'm not in the dead band

Consider the logical sequence of events as I drive the lead screw:

1. The stepper motor is turning the leadscrew such that motion is in one direction. The leadscrew is in contact with the nut.
2. The g-code tells the CNC software to change direction.
3. The stepper rotates in the opposite direction and begins to drive the leadscrew the equivalent of the backlash value. For example, if the leadscrew has a pitch of 0.100 inches and backlash was set to 0.025 inches, then the stepper would rotate one quarter revolution.
4. After the software has driven the stepper the specified backlash distance, it then starts moving the distance defined in the g-code.

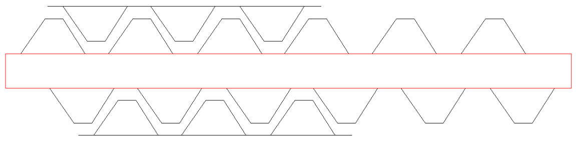
If there is no flexure in the system, as the stepper rotates the leadscrew its first step in the opposite direction, contact is broken between leadscrew and nut. In my system, this is at 0.00002 inches. But the system does have flexure so it will take some (hopefully) small but unknown number of steps to break the circuit.

Say I have a circuit that pulses the stepper at a high rate when the software is not pulsing and contact between leadscrew and nut is broken. See how the above sequence changes.



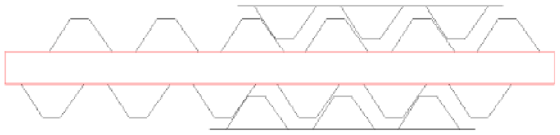
1. The stepper motor it turning the leadscrew such that motion is in one direction. The leadscrew is in contact with the nut.

2. The g-code tells the CNC software to change direction.



3. The stepper rotates in the opposite direction and begins to drive the leadscrew the equivalent of the backlash value.

We enter the dead band so contact between leadscrew and nut is broken. This cause my circuit to send pulses to the stepper at a high rate. It stops sending pulses when software sends a pulse. When we have traversed the dead band, contact between nut and leadscrew is again made. Pulse from the circuit then stop.



4. ~~Ideally,~~ after the software has driven the stepper the specified backlash distance, it then starts moving the distance defined in the g-code.

Movement due to the backlash compensation software function after the circuit has driven us through the dead band is overshoot. This value depends mostly on the flexure of the system.

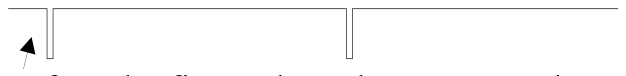
Going into more detail, consider the pulses seen by the stepper motor driver.

Assume no flexure in the system for now. I set the backlash to 0.000 04 inches which means only two pulses are sent to the motor driver when the software is told to change direction. Set the speed to 0.3 IPM. Given a maximum speed of 15 IPM, I set my percentage to $\frac{0.3}{15} \times 100\% = 2\%$.

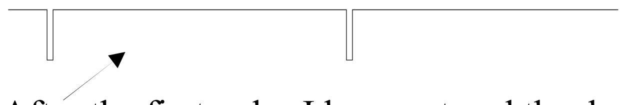
I can then calculate the pulse rate due to software:

$$\frac{0.3\ inches}{minute} \times \frac{1\ minute}{60\ seconds} \times \frac{1\ pulse}{0.00002\ inches} = \frac{1\ pulse}{4\ milli\ seconds} \quad (1)$$

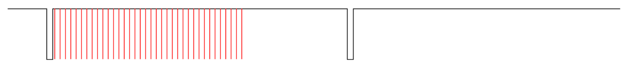
Since my backlash was set to 0.000 04 inches, I will get a total of two pulses from the software.



Before the first pulse, I have contact between nut and leadscrew.



After the first pulse I have entered the dead band and no contact is present. This tells the circuit to start pumping out pulses.



Say my worst case backlash is 0.004 inches. I want to traverse this distance in 4 milliseconds[3]. What pulse rate must the circuit generate?

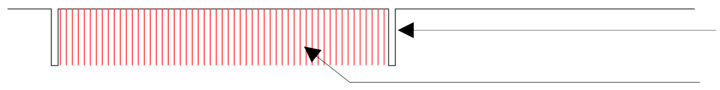
$$\frac{0.004\ inches}{0.004\ seconds} \times \frac{1\ pulse}{0.00002\ inches} = \frac{50{,}000\ pulse}{seconds} \quad (2)$$

This drives the leadscrew at a rate of

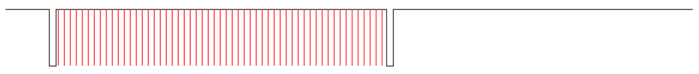
$$\frac{0.004\ inches}{0.004\ seconds} \times \frac{60\ seconds}{minute} = \frac{60\ inches}{minute} \quad (3)$$

Note that this fast rate is only occurring while in the dead band so there is only a small load on the motor.

[3] The pulse width is 5 microseconds so it can be ignored when compared to 4 milliseconds.

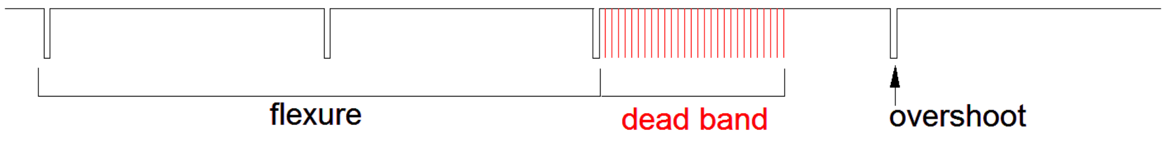


At or before the second software generated pulse, the circuit stops injecting pulses.



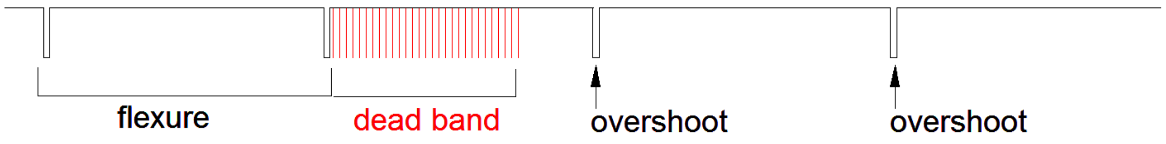
The software has been told to output two pulses. The first pulse breaks contact between leadscrew and nut. The second pulse occurs after we exit the dead band so is overshoot. In this ideal case of no flexure, the overshoot is only a single pulse so a mere 0.000 02 inches.

Next, assume we have flexure but it is a constant value. The backlash value in software must be the sum of the flexure and dead band.



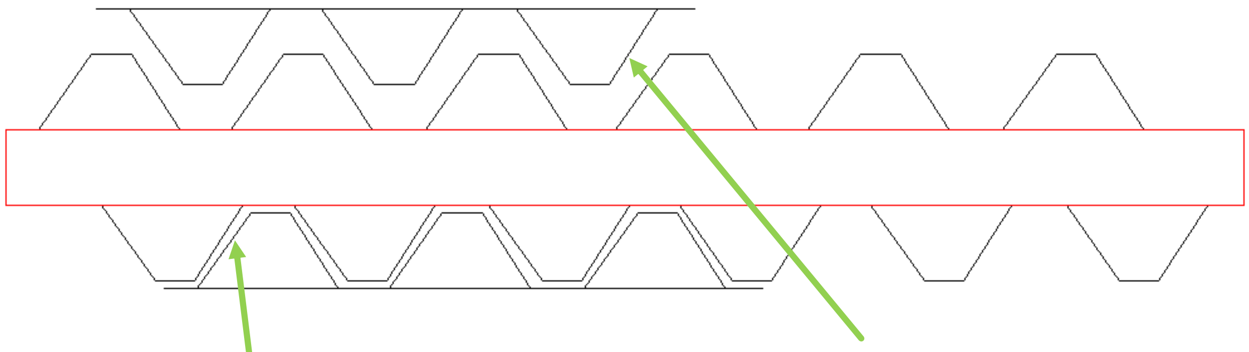
Some number of software generated pulses come from the backlash function rotating the leadscrew until we finally reach the dead band. Then the circuit starts up, moves us quickly through the dead band and stops. The next pulse from software is the last one and is our overshoot.

More likly, the flexure is not constant. We could end up with

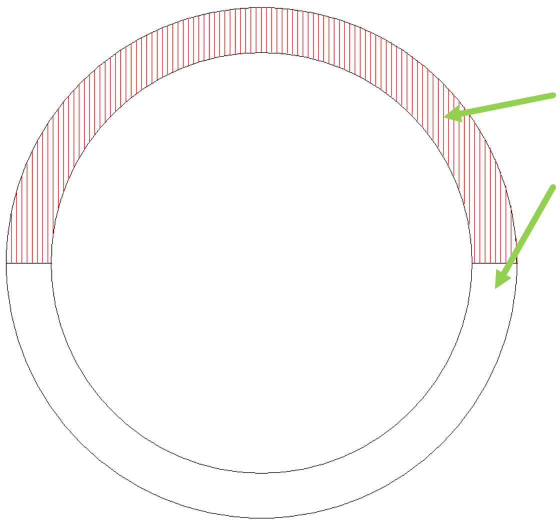


In this case, the flexure was less than expected. Those software generated pulses intended to move us beyond flexure and into the dead band end up as overshoot. If the flexure varies by 0.000 5 inches, we can expect this much overshoot. That might be acceptable. But if the flexure varies by 0.005 inches, an overshoot of 5 thou erases any value in the Adaptive Backlash Reduction scheme. Of course, this much variability in a mill is in general bad news.

Given a "reasonably small" change in flexure, this scheme should work. However, I ran into a gotcha -



The leadscrew tends to rest on the nut. So while the top side threads don't touch,
the bottom ones do. The change in resistance as I move from contact into the dead band is on the order of a few tens of milliohms and is not consistent. I was unable to align the leadscrew accurate enough to be centered in the nut.



One possible solution is to build a nut that has bronze threads on the top and
acetyl threads on the bottom. The insulating acetyl would then cradle the leadscrew while the bronze threads carried the load plus provided dead band information.

It sure would be nice to hear about a better solution!

Electronics

The electronics needed here consists of a filter and an oscillator. The filter ignores short breaks in continuity between leadscrew and nut. It quickly passes the fact that the leadscrew and nut contact. In this way any insulating gung on the threads is ignored. An opto isolator prevents machine ground noise from entering the oscillator's ground.

The oscillator generates a continuous 50 KHz pulse stream. It is OR'd with the software generated pulses only when there has not been contact between leadscrew and nut for a period of time set in the filter.

Acknowledgments

Thanks to John Herrmann for collaborating with me.

I welcome your comments and questions.

If you wish to be contacted each time I publish an article, email me with just "Article Alias" in the subject line.

Rick Sparber
Rgsparber.ha@gmail.com
Rick.Sparber.org